ABC INFORMATIQUE

N° 38

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

- Le traitement de texte
- L'Atmos de Oric
- Jeux à entendre
- Le basic français

EDITIONS ATLAS

M6062-38-12F 85FB-3,80FS-$1.95

aBC INFORMATIQUE

cours d'informatique pratique et familiale

EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 28-10-10. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS, Z.I. de Lucé, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume IV, n° 38.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Remy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Claire Bischoff *(correction)*.
Crédit photographique, couverture : SMT-Goupil.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : octobre 1984. 58410. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Traitement de texte

Le traitement de texte est une application très populaire sur les micro-ordinateurs. Il facilite grandement l'exécution des tâches comme produire des lettres ou des rapports.

Les systèmes de traitement de texte ne sont en fait que des machines à écrire informatisées. Le texte est tapé au clavier et apparaît à l'écran. Les modifications peuvent être apportées facilement sans avoir à retaper complètement le document. Lorsque le texte est satisfaisant, il peut être imprimé au moyen d'une imprimante.

Outre la crainte qu'inspire souvent l'ordinateur, ce qui éloigne nombre de personnes du traitement de texte est la certitude de ne pas taper assez rapidement sur le clavier. Il est pourtant beaucoup plus facile de commencer à utiliser un clavier sur un tel système plutôt que sur une machine à écrire ordinaire. Les inévitables erreurs répétées du dactylographe débutant sont très frustrantes. Ces erreurs peuvent être corrigées en quelques secondes avec un système de traitement de texte et la copie produite est parfaite à la première tentative.

Presque tous les micros domestiques peuvent effectuer du traitement de texte, mais certains sont mieux adaptés que d'autres pour cette tâche. Le coût d'un système de traitement de texte, même simple, peut être assez élevé, puisque les périphériques nécessaires peuvent coûter jusqu'à deux fois le prix de l'ordinateur.

Même les imprimantes les plus simples sont assez coûteuses et la qualité d'impression obtenue est relativement médiocre alors que, dans de nombreux cas, le traitement de texte exige une excellente qualité. Les imprimantes à marguerite offrent une meilleure qualité d'impression mais sont lentes et encore chères, bien qu'aujourd'hui les prix diminuent rapidement. Certaines machines à écrire électroniques peuvent être équipées d'interfaces qui permettent de les utiliser comme imprimantes.

Il est également possible pour un groupe d'amis ou pour un club d'informatique de partager le coût d'achat d'une imprimante de qualité. Les utilisateurs auront à résoudre le problème de l'interface entre leurs micros et l'imprimante. Pour certains ordinateurs munis d'interface standard, ce problème se résout facilement. L'utilisateur n'a qu'à acheter un câble approprié pour relier les deux unités.

Après avoir acquis une imprimante adéquate, votre tâche suivante consiste à choisir le programme de traitement de texte approprié. Une vaste gamme de logiciels est produite pour les marques les plus populaires d'ordinateurs, tandis que seulement un ou deux programmes peuvent être proposés pour des machines moins répandues. La qualité varie considérablement d'un programme à l'autre. Certains programmes sont rudimentaires et ne permettent qu'une simple édition, comme l'insertion et l'effacement de texte. D'autres permettent de déplacer des passages entiers à l'intérieur d'un article, de présenter le texte exactement comme il sera imprimé sur le papier, ou de justifier le texte (ajuster l'espacement entre les mots afin que la

Micro BBC
La combinaison d'un micro BBC et d'une unité de disquettes Torch autorise l'utilisation de logiciels professionnels, comme Wordstar. Le coût du système est cependant plus élevé que celui de nombreux micros professionnels (plus de 30 000 francs). Une imprimante moins coûteuse aurait peut-être pu être utilisée, et le logiciel de traitement de texte fourni avec l'unité de disquettes Torch être employé au lieu d'acheter Wordstar. Cela aurait réduit le prix à environ 18 000 F. (Cl. Ian McKinnell.)

longueur des lignes soit uniforme, comme pour le texte que vous lisez actuellement).

Certains programmes de traitement de texte peuvent rechercher un mot ou une phrase particulière. Ainsi une faute d'orthographe qui a été répétée tout au long d'un article peut être facilement corrigée. Il existe également des programmes de vérification d'orthographe pour certains systèmes. D'autres programmes de traitement de texte plus sophistiqués peuvent utiliser les caractéristiques évoluées de certaines imprimantes. Les imprimantes matricielles peuvent souvent produire plusieurs types de caractères (comme des caractères italiques ou caractères gras) et certains systèmes de traitement de texte permettent d'utiliser ces caractères dans un document. Quelques systèmes peuvent réussir à commander la production de graphiques sur certaines imprimantes matricielles. Cela permet d'utiliser plusieurs nouvelles polices de caractères, comme des grosses lettres ou des caractères gothiques. Utilisée judicieusement, la présentation d'un texte peut être grandement améliorée par un bon emploi de ces différentes possibilités.

Les imprimantes à marguerite peuvent utiliser « l'espacement proportionnel » pour accorder plus d'espace aux lettres larges comme le « w » et moins d'espace aux lettres étroites comme le « i », au lieu d'accorder des espaces égaux comme le ferait une machine à écrire ordinaire. Certains systèmes utilisent cette caractéristique, ce qui rend le texte beaucoup plus lisible, et permet de « justifier » les deux marges. C'est idéal pour produire des journaux ou des publications d'associations, puisqu'il est ainsi possible d'obtenir une présentation d'aspect professionnel sans avoir à supporter les coûts de la photocomposition. Les roues d'impression interchangeables permettent de choisir divers types de caractères selon les articles que l'on désire rédiger. Les logiciels de traitement de texte sont vendus sous divers formats : sur bande, sur disquette, en cartouche et sur puce ROM. Le plus important est cependant le mode de stockage du texte — généralement sur bande ou sur disquette. Bien qu'économique, le stockage sur bande est peu commode, lent, et il limite la longueur des documents à la dimension de la mémoire. Les disquettes sont préférables parce qu'elles sont rapides, fiables et qu'elles permettent d'écrire de longs documents. De nouveaux modes de stockage de données commencent à apparaître. Le Microdrive Sinclair, par exemple, est peu coûteux et peut malgré tout stocker de grandes quantités de données et trouver une partie spécifiée du texte en quelques secondes. Cependant, il n'existe encore que très peu de programmes de traitement de texte capables de travailler avec le Microdrive. Le lecteur de bande utilisé par le Coleco Adam (un micro domestique destiné particulièrement au traitement de texte puisqu'il comprend une imprimante à marguerite) est un autre système intéressant. Il utilise des bandes cassette modifiées comme support de stockage et peut trouver tout élément de données en quelques secondes.

Sauvegarder une copie du texte traité sur bande ou sur disquette permet d'écrire de longs articles sur plusieurs jours, d'utiliser plusieurs fois des lettres types et de copier tout travail devant être conservé. Il est aussi conseillé de faire des copies d'un long document à divers stades de sa rédaction. Cela représente une sécurité qui permet de ne pas perdre totalement un travail lors d'un incident quelconque.

Certains programmes comportent des commandes bizarres et des combinaisons de touches difficiles à mémoriser, tandis que d'autres sont plus simples à utiliser. Les claviers munis de touches de fonction sont préférables parce qu'ils diminuent le nombre de codes de commande devant être mémorisés. L'affichage peut aussi être une source de problèmes. Certains

Sinclair Spectrum
Plusieurs contraintes sont imposées par le Spectrum, dont un clavier de mauvaise qualité, l'absence d'une interface de moniteur et l'utilisation de Microdrive et non de lecteurs de disquettes. Il est cependant possible d'ajouter un meilleur clavier. Tasword Two est l'un des rares programmes de traitement de texte du Spectrum qui fonctionnent avec les Microdrive. L'ensemble revient à près de 7 000 francs. (Cl. Ian McKinnell.)

micros n'affichent qu'une portion du texte, ce qui complique beaucoup l'écriture. Le Vic-20 de Commodore, par exemple, ne peut afficher qu'un texte de 22 caractères de largeur, tandis que la largeur d'affichage des machines professionnelles atteint 80 caractères. Pour une utilisation intensive, l'idéal serait un affichage de 25 par 80 caractères sur un véritable moniteur afin d'obtenir une image claire et précise.

La conception des caractères affichés varie considérablement. Certaines machines utilisent plus de points que d'autres pour former chaque caractère, ce qui rend l'affichage plus facile à lire et à utiliser. Quelques micros utilisent des lettres formées par une matrice de six points par six tandis que la plupart disposent d'une grille huit points par huit. Quelques machines professionnelles offrent une remarquable qualité d'affichage avec des caractères de 16 points par 16.

Des systèmes domestiques modestes permettent l'écriture de lettres ou d'autres documents, mais un équipement plus évolué doit être utilisé pour écrire des livres ou de longs rapports. Si vous devez faire un travail intensif, vous avez besoin d'un système muni d'un moniteur, de deux lecteurs de disquettes, d'une bonne imprimante, d'un clavier de type machine à écrire et d'un excellent logiciel de traitement de texte. Transformer un micro domestique à ce niveau est très coûteux — souvent même plus coûteux que l'achat d'un véritable micro professionnel.

Les machines professionnelles offrent d'autres avantages. Puisqu'elles sont conçues pour des besoins professionnels, elles possèdent d'excellents claviers, écrans, lecteurs de disquettes et interfaces d'imprimante. Leur principal avantage est l'excellente qualité des logiciels dont elles disposent. Il existe pour ces machines un vaste choix de bons logiciels parce que la plupart des ordinateurs professionnels utilisent l'un des systèmes d'exploitation standard, ce qui signifie que chaque programme de traitement de texte peut être utilisé par de nombreuses machines différentes.

Wordstar est de loin le logiciel de traitement de texte le plus connu et il est disponible pour les systèmes d'exploitation CP/M, CP/M-86 et MS-DOS. Le programme a quelques fonctions sophistiquées mais il coûte assez cher, plus de vingt fois le prix d'un programme de micro domestique. Le prix d'un système de traitement de texte professionnel est élevé. Mais utiliser un micro domestique pour un traitement de texte intensif est une fausse économie pour une petite entreprise ou pour un écrivain professionnel. Le temps perdu en utilisant un système peu évolué annulera rapidement l'argent épargné.

Le coût d'un système professionnel sérieux est élevé, mais il diminue tous les jours. Certains micros domestiques peuvent être étendus pour utiliser des systèmes d'exploitation standard comme CP/M. Ainsi, les gens qui ont déjà beaucoup investi dans un système domestique commencent à être en mesure d'utiliser un logiciel sérieux sans avoir à payer une somme supplémentaire trop importante. Une autre tendance aide actuellement à faire baisser le prix des bons programmes. Plusieurs sociétés offrent gracieusement des programmes de traitement de texte comme Wordstar avec leurs ordinateurs. Mais attention, des constructeurs donnent des programmes qui ne sont pas au niveau de Wordstar.

Dernière nouveauté en la matière : le traitement de texte mobile. Plusieurs ordinateurs alimentés par pile sont vendus avec des programmes de traitement de texte intégrés, ce qui permet de composer des documents en tout temps et en tout lieu. Ces machines ne se prêtent pas à de nombreux autres usages et elles sont hors de portée du budget des utilisateurs de micros domestiques. Mais la présence de ce type de machine laisse présager que le traitement de texte deviendra de plus en plus populaire.

Commodore 64
Le Commodore 64 offre le système de traitement de texte avec lecteur de disquettes le moins cher sur le marché. Le fait de ne disposer que d'un seul lecteur de disquettes est une contrainte en soi et le Commodore 64 ne peut produire qu'un affichage d'une largeur de 40 caractères. Une imprimante moins coûteuse est offerte à environ 2 400 francs mais la qualité d'impression est médiocre. L'ensemble avoisine les 10 000 francs. (Cl. Ian McKinnell.)

Laisser une empreinte

La création de plans objets est l'élément le plus intéressant des possibilités graphiques du Commodore 64, parce qu'il permet d'écrire en basic des jeux à action rapide.

Un plan objet est un motif graphique mobile. Il est conçu un peu comme sur le système des « huit par huit » définis par l'utilisateur, déjà traité précédemment. Mais il est construit sur une grille beaucoup plus grande. Dès que le plan objet est défini, les attributs comme la couleur et la position sur l'écran peuvent être commandés par l'ensemble des registres de la puce de commande vidéo du Commodore 64 (la puce VIC).

Un plan objet est formé par 21 lignes de 24 points. Chaque ligne est composée de trois segments de huit points et est représentée par trois octets en mémoire; il faut donc utiliser 63 octets pour stocker les données d'un plan objet. Chaque point de la grille du plan objet est donc représenté par un 1 binaire (et les points non illuminés par un 0 binaire). Nous devons donc calculer les équivalents décimaux de chaque groupe de huit chiffres binaires.

Dès qu'un plan objet a été défini et converti en une série d'instructions DATA, celles-ci doivent être lues (READ) et écrites (POKE) en mémoire. Les données peuvent être positionnées en plusieurs emplacements mémoire. Par exemple, utiliser les adresses 12288 et suivantes les placerait dans la zone du programme BASIC, qui va de 2048 à 40960.

Lors de la saisie d'un programme BASIC, le programme remplit l'espace mémoire à partir de l'adresse 2048 et suivantes.

Un programme doit avoir une longueur de 10 K pour atteindre l'adresse 12288 et ainsi écraser les données des plans objets. Cependant, lors de l'exécution, toute variable utilisée est stockée dans la zone située au-dessus de celle employée pour stocker le programme — les variables de chaîne, par exemple, sont situées au haut de la zone de programme BASIC. Comme le jeu Chasse aux sous-marins utilise la variable de minuterie, TI$, la mise à jour régulière de sa valeur et son stockage subséquent écraseraient éventuellement la zone dans laquelle nous désirons stocker les données des plans objets.

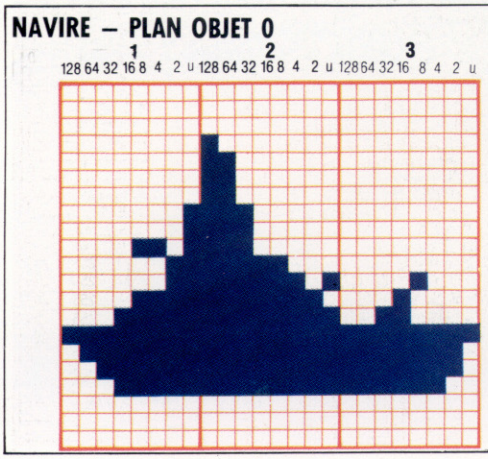
Ce problème peut être résolu en abaissant le haut de la zone du programme BASIC sous la zone où sont stockées les données des plans objets. Le pointeur de l'adresse du haut de la mémoire est stocké aux adresses 55 (octet bas) et 56 (octet haut). Normalement, ces deux adresses renferment respectivement les valeurs 0 et 160, ce qui représente l'adresse 40960. Sous cette forme bas-haut, l'adresse 12288 est définie par les valeurs 0 et 48. Nous pouvons abaisser le haut de la mémoire en écrivant simplement ces valeurs dans les adresses 55 et 56 au début du programme.

Les pointeurs de plans objets

Comme les données des plans objets peuvent être placées dans diverses parties de la mémoire, il est nécessaire d'utiliser un pointeur pour indiquer où commencent les données. Il y a huit pointeurs de plans objets, contenus dans les adresses 2040 (plan objet 0) à 2047 (plan objet 7). La valeur contenue dans chaque pointeur désigne la zone qui renferme les données de plans objets selon la formule : début des 63 octets de données = (pointeur des plans objets) × 64. Les données relatives au navire de notre programme commencent à 12288 et le navire sera le plan objet 0, le pointeur de l'adresse 2040 est donc 192 (12288/12). Le pro-

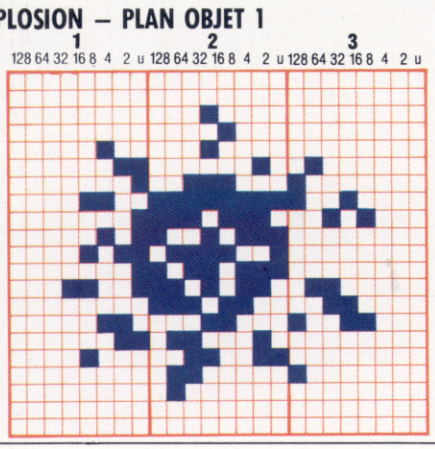
Cible mobile
Un plan objet est composé de 21 lignes de trois octets; ces octets sont en fait des configurations binaires qui sont stockées dans les instructions de données du programme BASIC sous la forme de leurs équivalents décimaux. Ces valeurs apparaissent dans les diagrammes de conception et dans le listage du programme. Le programme écrit les valeurs dans une zone dédiée de la mémoire, où la puce de commande vidéo les interprète comme données de plan d'objet, les affiche et les déplace sur l'écran en demandant un minimum d'effort de programmation.

NAVIRE – PLAN OBJET 0



1	2	3
0	0	0
0	0	0
0	0	0
0	128	0
0	192	0
0	192	0
0	192	0
1	224	0
1	224	0
13	224	0
3	248	128
3	253	8
15	254	16
31	255	48
255	255	255
127	255	254
63	255	254
31	255	252
0	0	0
0	0	0
0	0	0

EXPLOSION – PLAN OBJET 1



1	2	3
0	0	0
0	0	0
0	16	0
0	8	0
4	16	0
3	2	64
1	56	128
12	255	144
1	238	40
5	151	0
11	121	0
1	183	0
25	214	96
0	236	48
6	24	152
3	98	0
8	51	0
0	96	128
0	64	0
0	0	0
0	0	0

chain bloc est destiné à l'explosion, le plan objet 1. Si nous donnons au pointeur situé en 2041 la valeur de 193, les données doivent commencer à 12352. Voici les valeurs que nous utilisons :

Numéro de plan objet	0	1	2	3
Pointeur de plan objet	192	193	194	195
63 octets de données de plans objets	12288 à 12350	12352 à 12414	12416 à 12478	12480 à 12542

Notez qu'un octet demeure inutilisé à la fin de chaque bloc de données. Les parties du listage qui lisent les données de plan objet et qui spécifient les pointeurs de plan objet sont comprises entre les lignes 2000 et 2210.

Manipulation des plans objets

La puce de commande vidéo (VIC) possède plusieurs registres spéciaux qui servent à commander les plans objets. La première adresse de la puce VIC est 53248 et, pour simplifier notre propos, cette adresse servira de point de référence lors de la description des autres registres. Si V=53248, l'adresse suivante peut être désignée par l'expression V+1 et ainsi de suite. V doit être défini, comme d'autres variables, au début du programme (voir ligne 100).

La couleur de chaque plan objet est définie en écrivant un code de couleur (de 0 à 15) dans un registre spécial. Chacun des huit plans objets a son propre registre de couleur; ceux-ci vont de V+39 à V+46. Par exemple, pour colorer le navire en noir nous n'avons qu'à écrire (POKE) le code de couleur 0 dans l'adresse V+39. Les autres plans objets peuvent être colorés de la même manière (voir les lignes 2220 à 2250).

Le positionnement des différents plans objets sur l'écran sera expliqué de façon détaillée plus loin.

Les plans objets peuvent être agrandis horizontalement et verticalement d'un facteur 2. Les plans objets du navire et du sous-marin sembleront plutôt écrasés horizontalement, mais nous doublerons leur longueur initiale. En fait, les quatre plans objets seront agrandis horizontalement. Le registre de la puce VIC commandant l'extension horizontale est V+29, plus facile à utiliser que les autres registres que nous avons présentés jusqu'ici. Au lieu d'utiliser huit registres différents pour commander les attributs de chacun des huit plans objets, il suffit ici de valider ou d'interdire cette fonction. Par conséquent, seul un bit de ce registre suffit à commander l'extension horizontale de chaque plan objet. Si un plan objet doit être étendu horizontalement, le bit correspondant du registre V+29 doit être mis à 1. Le tableau suivant illustre quel doit être le contenu de V+29 pour allonger les quatre plans objets que nous avons définis :

Numéro de plan objet	7	6	5	4	3	2	1	0
Contenu de V+29	0	0	0	0	1	1	1	1

= 15 (décimal)

L'extension dans le sens vertical est commandée par le registre V+23. L'explosion, plan objet 1, est étendue verticalement et horizontalement, ce qui double en fait ses dimensions (voir les lignes 2290 à 2310).

Notre dernière tâche est d'illuminer les plans objets appropriés. Un seul bit du registre V+21 de la puce VIC sert à afficher ou à éteindre les plans objets. Dans notre jeu, seuls le navire et le sous-marin sont initialement affichés (lignes 2310 à 2360).

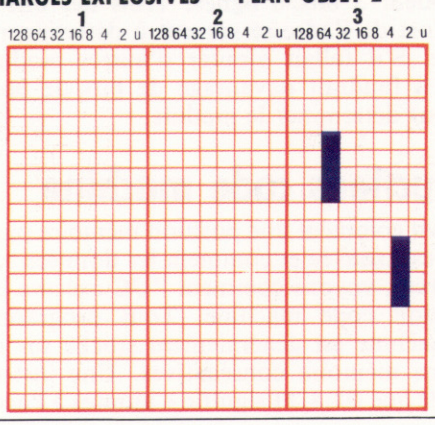
Une fois tapé tout le listage, vous devriez tester l'exactitude des données des plans objets. Pour ce faire, exécutez le programme et interrompez-le à l'aide des touches RUN et STOP.

```
POKE V,160          (Coordonnées du navire)
POKE V+2,240:       (Coordonnées x et y
  POKE V+3,100      de l'explosion)
POKE V+4,160:       (Coordonnées x et y
  POKE V+5,100      de la charge explosive)
POKE V+6,100:       (Coordonnées x et y
  POKE V+7,100      du sous-marin)
POKE V+21,15        (Affiche les plans objets 0-3)
```

Si le programme s'arrête en produisant le message "OUT OF DATA ERROR", vérifiez combien de nombres renferment les instructions DATA. Ce devrait être 63 pour chaque plan objet. Si le programme s'arrête anormalement et si le clavier ne répond plus, assurez-vous que la variable V a été déclarée à la ligne 100. Il est préférable de sauvegarder votre programme avant de l'exécuter.

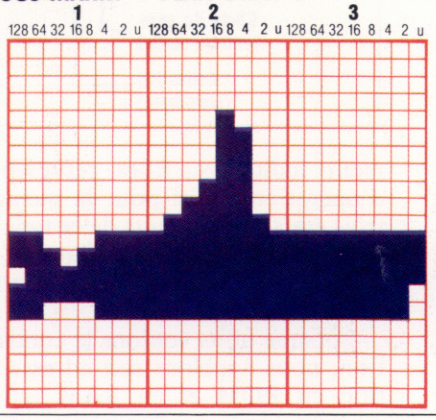
```
1 REM***GRAPHIQUES C64********
90 POKE 55,0:POKE 56,48:CLR
:REM LOWER MEMTOP
100 V=53248:FL=0:SC=0
110 GOSUB 1000
:REM PREPARATION D'ECRAN
120 GOSUB 2000
:REM CREATION DE PLAN OBJET
1000 REM**CREATION DE PLAN OB
JET                          **
1020 REM**LECTURE DONNEES NAV
IRE                          **
1030 FOR I=12288 TO 12350
1040 READ A:POKE I,A:NEXT I
1060 REM**LECTURE DONNEES EXP
                             **
1070 FOR I=12352 TO 12414
1080 READ A:POKE I,A:NEXT I
1100 REM**LECTURE DONNEES CHG
                             **
1110 FOR I=12416 TO 12478
1120 READ A:POKE I,A:NEXT I
1140 REM**LECTURE DONNEES S-M
                             **
1150 FOR I=12480 TO 12542
1160 READ A:POKE I,A:NEXT I
1180 REM**DEF POINTEURS      **
1190 POKE 2040,192:POKE 2041,
193:POKE 2042,194:POKE
2043,195
2220 REM**DEF COULEURS       **
2230 POKE V+39,0:POKE V+40,1:
POKE V+41,0:POKE V+42,0
2260 REM**COORD INIT NAVIRE
                             **
2270 POKE V+1,80:X0=160      **
2290 REM**EXT PLANS OBJETS
2300 POKE V+29,15:POKE V+23,2
2320 REM**ALLUMER PLANS OBJETS
                             **
2330 POKE V+21,9
2340 RETURN
2350:
6000 REM**DONNEES NAVIRE     **
6010 DATA 0,0,0,0,0,0,0,0
6020 DATA 0,128,0,0,192,0,0,
192,0
6030 DATA 0,192,0,1,224,0,1,
224,0
6040 DATA 13,224,0,3,248,128,
3,253,8
6050 DATA 15,254,16,31,255,48,
255,255,255
6060 DATA 127,225,254,63,255,
254,31,255,252
6070 DATA 0,0,0,0,0,0,0,0
6100 REM**DONNEES EXPLOSION
                             **
6110 DATA 0,0,0,0,0,0,16,0,
0,8,0,4,16
6120 DATA 0,3,2,64,1,56,128,
12,255,144
6130 DATA 1,238,48,5,151,0,11,
121,0,1
6140 DATA 183,0,25,214,96,0,
236,48,6,24
6150 DATA 152,3,98,0,8,51,0,
0,96,128,0
6160 DATA 64,0,0,0,0,0,0,0
6200 REM**DONNEES PROFONDEUR C
HG                           **
6210 DATA 0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
6220 DATA 0,0,0,32,0,0,32,0,
0,32,0,0,32,0
6230 DATA 0,0,0,0,0,0,0,0
6240 DATA 2,0,0,2,0,0,2,0,0,
2,0,0
6250 DATA 0,0,0,0,0,0,0,0
6260 DATA 0,0,0,0,0,0,0,0
6300 REM**DONNEES S-M        **
6310 DATA 0,0,0,0,0,0,0,0,0,
0,0,0
6320 DATA 0,0,0,0,12,0,0,12,0
6330 DATA 0,12,0,0,28,0,0,60,0
6340 DATA 0,126,0,199,255,255
6350 DATA 239,255,255,127,
255,255
6360 DATA 255,255,254,199,
255,254
6370 DATA 0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
```

CHARGES EXPLOSIVES – PLAN OBJET 2



1	2	3
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	32
0	0	32
0	0	32
0	0	32
0	0	0
0	0	0
0	0	2
0	0	2
0	0	2
0	0	2
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

SOUS-MARIN – PLAN OBJET 3



1	2	3
0	0	0
0	0	0
0	0	0
0	0	0
0	8	0
0	12	0
0	12	0
0	12	0
0	28	0
0	60	0
0	126	0
199	255	255
239	255	255
127	255	255
255	255	254
199	255	254
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Liz Dixon

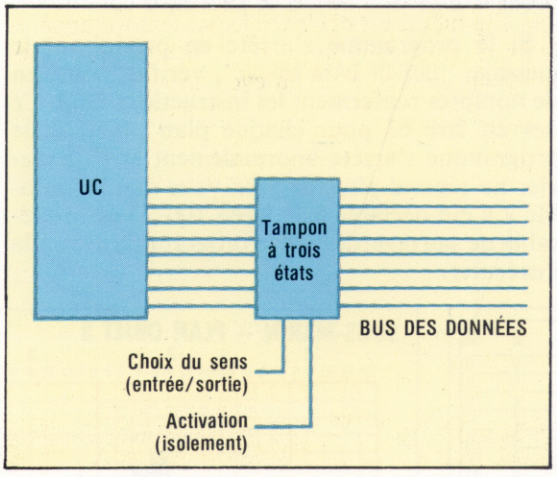
Parcours mémoire

L'Unité Centrale ou UC constitue le centre nerveux de votre ordinateur personnel. Nous verrons dans cet article la logique du transfert de données entre l'UC et la mémoire.

Chaque position mémoire d'un micro-ordinateur comprend habituellement 8 bits. Les données sont transférées à raison de 8 bits à la fois, via un ensemble de 8 lignes parallèles, vers l'UC où elles sont traitées par les instructions du programme. Les données peuvent également transiter dans l'autre sens afin d'être stockées sur des positions mémoire. Ainsi l'instruction code machine LDA $1234 envoie le nombre présent à la position $1234, vers l'UC par le canal du bus des données. STA $1234, pour sa part, transmet un nombre depuis l'UC sur la position mémoire $1234, par le même canal.

Le bus des données doit donc permettre le transfert de données dans les deux sens. Il est cependant parfois utile d'isoler l'UC du bus des données. A cette fin, chaque ligne de ce dernier peut être dans trois états (Input/Entrée, Output/Sortie et Isolate/Isolement). Pour effectuer les diverses commutations entre ces états, chaque ligne est dotée d'un petit circuit électronique appelé « périphérique à trois états ».

Un seul circuit intégré regroupe les 8 périphériques à trois états. Le diagramme ci-dessus montre la liaison UC-bus des données. Les lignes « choix du sens » et « activation » sont également indiquées. Elles représentent le mode d'opération choisi pour la mémoire tampon à trois états. Des circuits de ce type peuvent servir par ailleurs à connecter d'autres périphériques, comme des périphériques d'Entrée/Sortie, au bus des données.



Lorsque nous voulons le contenu d'une position, nous appelons son adresse. Chaque position en ROM et en RAM est référencée de manière univoque par une adresse. Étudions maintenant comment, au niveau matériel, une position est atteinte pour livrer les données qu'elle contient (transfert de données).

Outre le bus des données, la plupart des micro-ordinateurs comportent un autre canal de transmission, le « bus des adresses ». Ce dernier dispose habituellement de 16 lignes. Cela signifie qu'il est possible de spécifier jusqu'à 65 536 adresses (2^{16} = 65 536). Le bus à 16 bits des adresses permet donc d'accéder à 64 K de mémoire. L'ensemble de la zone mémoire peut être pensé en termes de modules la divisant et de 256 positions chacun. Les 8 bits de poids faible de l'adresse servent à trouver une position donnée d'un module déterminé. Le module est choisi par certains des bits restants, ou par tous.

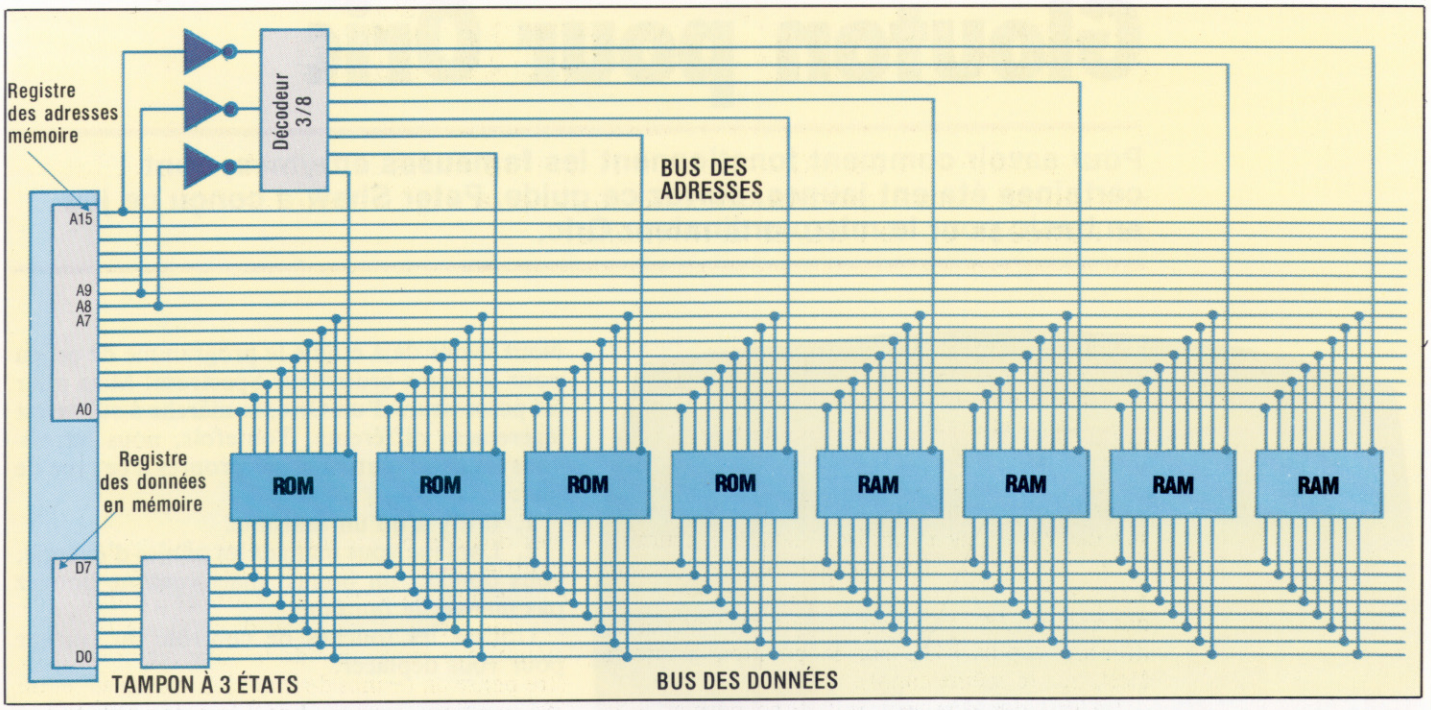
Si nous prenons l'exemple simple d'un micro-ordinateur doté d'une capacité mémoire totale de 2 K, nous pouvons étudier comment sont trouvées les positions mémoire. Un module mémoire comportant 256 positions, notre micro de 2 K prendra donc huit modules. Nous supposerons que la mémoire est divisée à parts égales entre RAM et ROM.

L'adresse de la position voulue figure dans un registre spécial à 16 bits de l'UC, appelé registre des adresses mémoire. Puisque les 8 bits de poids faible d'une adresse déterminent la position dans le module, les 8 lignes de poids faible du bus des adresses sont connectées aux 8 modules. Pour choisir un module, il suffit maintenant de trois bits supplémentaires (2^3 = 8). Ce code à trois bits doit être décodé en 8 lignes de sortie, une par module.

Le diagramme montre comment sont reliés avec l'UC les modules mémoire par l'intermédiaire des bus des adresses et des données. Chaque module de mémoire reçoit une ligne depuis le décodeur 3 bits/8 lignes. Trois des bits de poids fort de l'adresse servent à déterminer le module. S'il fallait ajouter d'autres modules RAM, il faudrait davantage de bits de poids fort pour déterminer le module.

Instructions code machine

Ayant vu comment une position mémoire particulière est trouvée et comment les données sont transférées, voyons comment l'UC exécute une instruction code machine. Tout programme code machine est généralement stocké sur des positions mémoire consécutives. Une instruction telle que ADD $13FF signifie « ajoutez à l'accu-

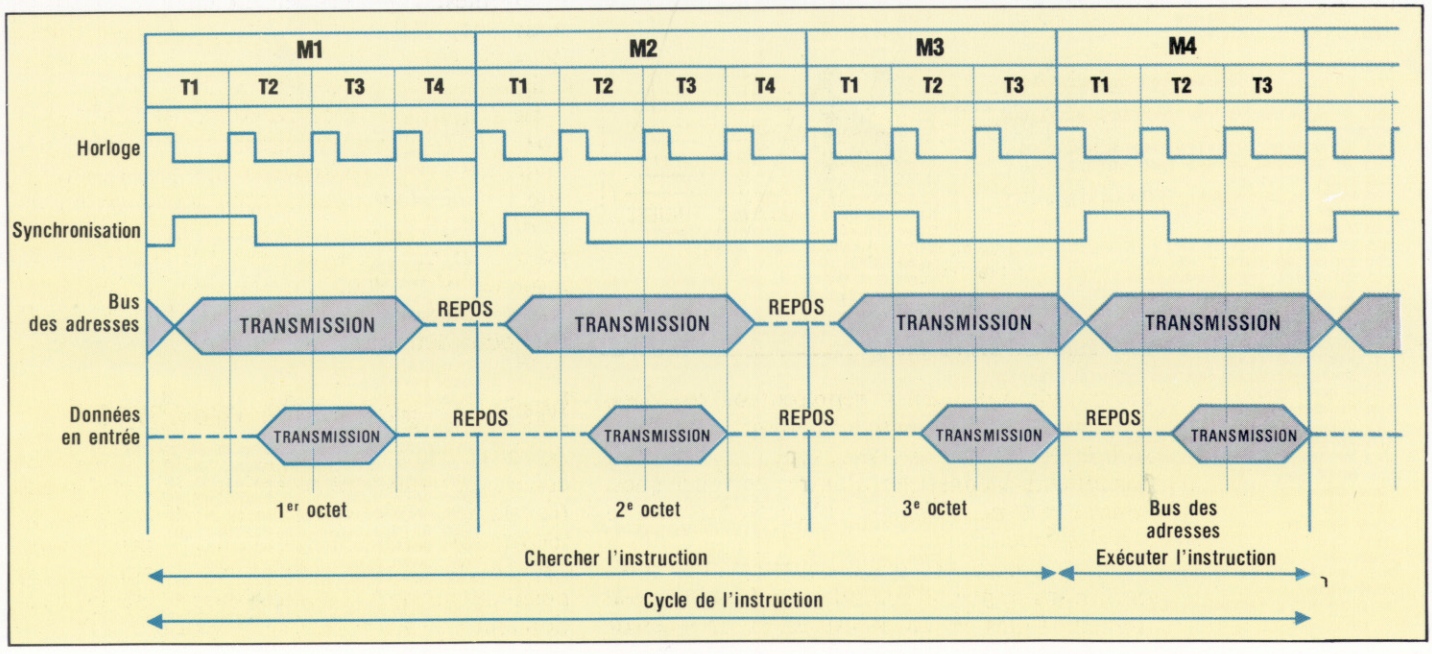


mulateur le contenu de la position d'adresse hexadécimale $13FF ». Cette instruction prendra 3 octets. Un pour le code binaire de l'instruction ADD, et deux pour l'adresse à 16 bits, $13FF. Supposons que les positions soient $1000, $1001 et $1002.

Avant que l'instruction puisse être exécutée, elle doit être sortie de la mémoire. Cela suppose trois accès distincts pour convoyer les trois octets par le bus des données vers l'UC. Après cela, l'instruction complète se trouve dans un registre spécial de l'UC. Dans notre exemple, l'instruction nécessite un accès mémoire supplémentaire pour aller chercher le contenu de la position mémoire $13FF de sorte qu'il soit ajouté à l'accumulateur.

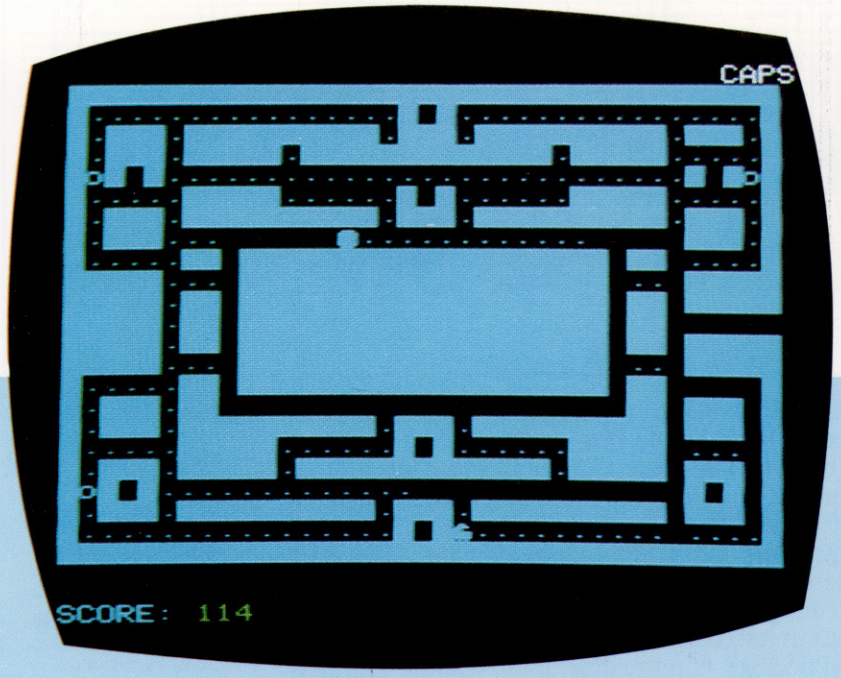
Les constructeurs de micro-ordinateurs indiquent les caractéristiques de leurs processeurs sous la forme de diagrammes de synchronisation. Ces derniers donnent l'ordre des événements pour les différentes opérations internes à l'ordinateur. Nous pouvons établir un diagramme de synchronisation pour les cycles « chercher » et « exécuter » d'une instruction code machine. La synchronisation des opérations est commandée à partir des impulsions d'horloge, et notre diagramme montre que dans le système que nous imaginons ici, le bus des adresses est activé par le bord avant de l'impulsion de synchronisation, alors que le bus des données est activé par le bord arrière. L'impulsion elle-même est déclenchée par le bord arrière de la première impulsion de l'horloge en phase de fonctionnement ou en cycle machine.

« Chercher » et « Exécuter » Une instruction code machine qui consiste en un octet code opération suivi de deux octets opérande est gérée au cours d'un cycle qui comprend les phases « Chercher » et « Exécuter ». Lors du premier cycle, le bus des adresses accède aux positions mémoire qui contiennent l'instruction, et le bus des données envoie les octets de l'instruction vers l'UC. Pendant le deuxième cycle, l'exécution de l'instruction met en œuvre les bus des données et des adresses. (Cl. Liz Dixon.)



Glouton pour Oric

Pour savoir comment fonctionnent les fameuses enzymes, dont certaines étaient jaunes, suivez ce guide. Peter Shaw a conçu ce jeu en basic pour le micro-ordinateur Oric.



Nous avons déjà étudié le programme de ce jeu pour le micro-ordinateur Spectrum. Mais pour les possesseurs d'un Oric, la marche à suivre est légèrement différente. Toutefois, nous retrouvons les deux particularités propres à ce type de jeu :

1. Il n'y a qu'un fantôme.
2. Lorsque vous mangez les pilules d'énergie, vous obtenez un bonus, mais vous ne pouvez pas manger le fantôme.

Utilisez les touches de contrôle du curseur pour vous déplacer.

```
30 RE=0:GOSUB 9000
40 GOSUB 8000
50 V%=19:H%=19:GOSUB 7000
60 PLOT H%,V%," "
70 A$=KEY$
80 A=DEEK(783)
90 REM
95 H1%=H%:V1%=V%
100 REM
110 IF A=48351 THEN H%=H%-1:M$="'"
120 IF A=48255 THEN H%=H%+1:M$="%"
130 IF A=48319 THEN V%=V%+1:M$="&"
140 IF A=48375 THEN V%=V%-1:M$="$"
150 IF H%<1 THEN H%=38
160 IF H%>38 THEN H%=1

165 PLAY 0,1,1,10
170 IF SCRN(H%,V%)=44 THEN V%=V1%:H%=H1%
:GOTO 210
180 IF SCRN(H%,V%)=46 THEN SC%=SC%+1:CO%
=CO%+1
190 IF SCRN(H%,V%)=111 THEN GOSUB 1000
200 IF SCRN(H%,V%)=43 THEN GOTO 5000
210 PLOT H%,V%,M$
220 PLOT 1,25,"SCORE:"+STR$(SC%)
230 PLOT X%,Z%,C$
240 Z1%=Z%:X1%=X%
245 IF M1%>3 THEN M1%=0
250 IF M1%=0 THEN Z%=Z%-1
260 IF M1%=1 THEN X%=X%-1
270 IF M1%=2 THEN Z%=Z%+1
280 IF M1%=3 THEN X%=X%+1
281 PLOT H%,V%,"#"
290 IF SCRN(X%,Z%)=44 THEN Z%=Z1%:X%=X1%
:M1%=INT(RND(1)*4)
300 IF SCRN(X%,Z%)<40 AND SCRN(X%,Z%)>33
 THEN GOTO 5000
310 C$=CHR$(SCRN(X%,Z%))
320 PLOT X%,Z%,"+"
330 IF LV%<1 THEN GOTO 2000
340 IF CO%=T% THEN CO%=0:GOTO 50
350 GOTO 60
1000 SC%=SC%+(INT(RND(1)*20))+20
1010 ZAP
1020 RETURN
2000 CLS
2005 ZAP
2010 PRINT,,,,"TERMINE"
2020 PRINT
2030 PRINT,,,"SCORE:";SC%
2040 PRINT
2050 PRINT
2060 IF SC%>RE THEN RE=SC%
2070 PRINT,,"RECORD ACTUEL:"RE
2080 PRINT
2090 PRINT
2100 PRINT,"TAPEZ UNE TOUCHE POUR JOUER"
2110 IF KEY$<>"" THEN GOTO 2120
2120 GET K$
2130 GOTO 40
5000 PLOT X%,Z%,"."
5010 PLOT H%,V%,"#"
5020 PLAY 1,0,1,10
5030 WAIT 15
5040 PLOT H%,V%,"("
5050 PLAY 1,0,1,20
5060 WAIT 20
5070 PLOT H%,V%,")"
5080 PLAY 1,0,1,25
5090 WAIT 30
5100 PLOT H%,V%,"*"
5110 PLAY 1,0,1,30
5120 WAIT 40
5130 PLOT H%,V%," "
5140 PLAY 1,0,1,50
5150 EXPLODE:WAIT 100
5160 CO%=0:V%=12:H%=16:LV%=LV%-1
5170 IF LV%<1 THEN 2000
5180 GOTO 50
7000 CLS:PAPER 0:INK INT(RND(1)*4)+3
7005 PLOT 0,25,6
7006 PING
7010 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7020 PRINT",,,,,,,,,,,,,,,,, ,,,,,,,,,,
,,,,,,,,"
7030 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7040 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7050 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7060 PRINT",,,,,,,,,,,,,,,,,, ,,,,,,,,,,
,,,,,,,,"
7070 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7080 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7090 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7100 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7110 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7120 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7130 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7140 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7150 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7160 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7170 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7180 PRINT",,,,,,,,,,,,,,,,,, ,,,,,,,,,,
,,,,,,,,"
7190 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7200 PRINT",,, ,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,, ,,,,"
7210 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7220 PRINT",,,,,,,,,,,,,,,,,, ,,,,,,,,,,
,,,,,,,,"
7230 PRINT",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,"
7500 RETURN
8000 M$="%"
8010 C1%=8:M1%=0
8020 SC%=0
8030 Z%=7:X%=19
8040 LV%=3
8050 CO%=0
8060 T%=312
8070 RETURN
9000 FOR U=(46080+(ASC("#")*8)) TO (4608
0+(ASC(",")*8)+7)
9010 READ US:POKE U,US:NEXT U:RETURN
9020 DATA 0,30,63,63,63,63,63,30
9030 DATA 0,18,51,51,63,63,30,12
9040 DATA 0,30,63,60,56,60,63,30
9050 DATA 0,12,30,63,63,51,51,18
9060 DATA 0,30,63,15,7,15,63,30
9070 DATA 0,0,0,0,63,63,63,30
9080 DATA 0,0,0,0,8,12,12,30
9090 DATA 0,33,18,12,0,12,18,33
9100 DATA 0,12,30,56,63,63,63,42
9110 DATA 63,63,63,63,63,63,63,63
```

Oric et son rejeton

L'Oric-1 fut lancé en 1983 sur le marché anglais, mais ne connut jamais un grand succès en raison de certaines erreurs de conception. Ces dernières ont été surmontées avec l'Oric Atmos.

Pourvu d'un puissant BASIC de type Microsoft, d'un port imprimante Centronics et d'une prise pour moniteur couleur de type RVB, l'Oric-1 avait bien des atouts pour séduire. Il fut pourtant victime — en Grande-Bretagne du moins — du faible nombre de logiciels disponibles et, surtout, de « bogues » très irritantes au niveau de la ROM.

Oric Products décida donc de remédier à ces faiblesses, et l'appareil, revu et modifié, est devenu l'Oric Atmos. Les anciennes touches, de type calculatrice, ont maintenant cédé la place à un véritable clavier; le boîtier a été redessiné. L'ensemble est d'un rouge et noir très élégant, et s'est vu adjoindre une touche de fonction supplémentaire, non encore connectée, mais qui doit rendre possibles certaines extensions futures.

L'Atmos est construit autour du microprocesseur 6502, et offre à l'utilisateur 37 K de RAM pour la programmation en BASIC. Il peut afficher huit couleurs différentes, et sa résolution maximale est de 240 × 200 pixels. Le jeu de caractères est installé en RAM, et peut donc être redéfini au gré de l'utilisateur, qui dispose également d'un second jeu de blocs graphiques de type Vidéotex. A la différence du Spectrum, qui possède un fichier particulier consacré aux caractéristiques d'écran, situé en RAM, l'Atmos fait usage de « caractéristiques en mode série », qui occupent moins d'espaces mémoire mais sont affichées sous formes vides; aussi l'affichage doit-il faire l'objet de soins particuliers.

Quatre sons prédéfinis sont enregistrés en ROM : ZAP, PING, SHOOT et EXPLODE. Ils sont destinés à la création de jeux d'arcade. La puce sonore de l'Oric est très sophistiquée, et des instructions comme MUSIC, SOUND et PLAY permettent la création de nombreux sons différents. Le volume sonore est réglable; trois voix et un canal bruit fournissent une étendue de sept octaves.

Le BASIC de l'Oric-1 était affligé de plusieurs bogues très gênantes. L'instruction TAB ne fonctionnait pas correctement, et les paramètres

Configuration d'ensemble
L'Oric Atmos est un ordinateur domestique de prix très raisonnable, pourvu de 48 K de mémoire, d'un graphisme couleur et de possibilités sonores. Oric produit deux périphériques, dotés des mêmes couleurs rouge et noir pour l'appareil. Le lecteur de disquettes permet une plus grande rapidité que les cassettes, et l'imprimante/table traçante peut écrire ou dessiner en quatre couleurs. (Cl. Ian McKinnell.)

Oric-1
L'Atmos est une version améliorée de l'Oric-1. Le circuit imprimé est le même, mais la puce ROM est différente, et abrite une variante de BASIC débarrassée de ses erreurs. Tout cela suffit à faire de l'Atmos une machine bien meilleure. Les possesseurs d'Oric-1 peuvent faire procéder à un échange en s'adressant à Oric-France. Cela leur coûtera environ 700 francs. Mais les logiciels ne sont pas tous compatibles.

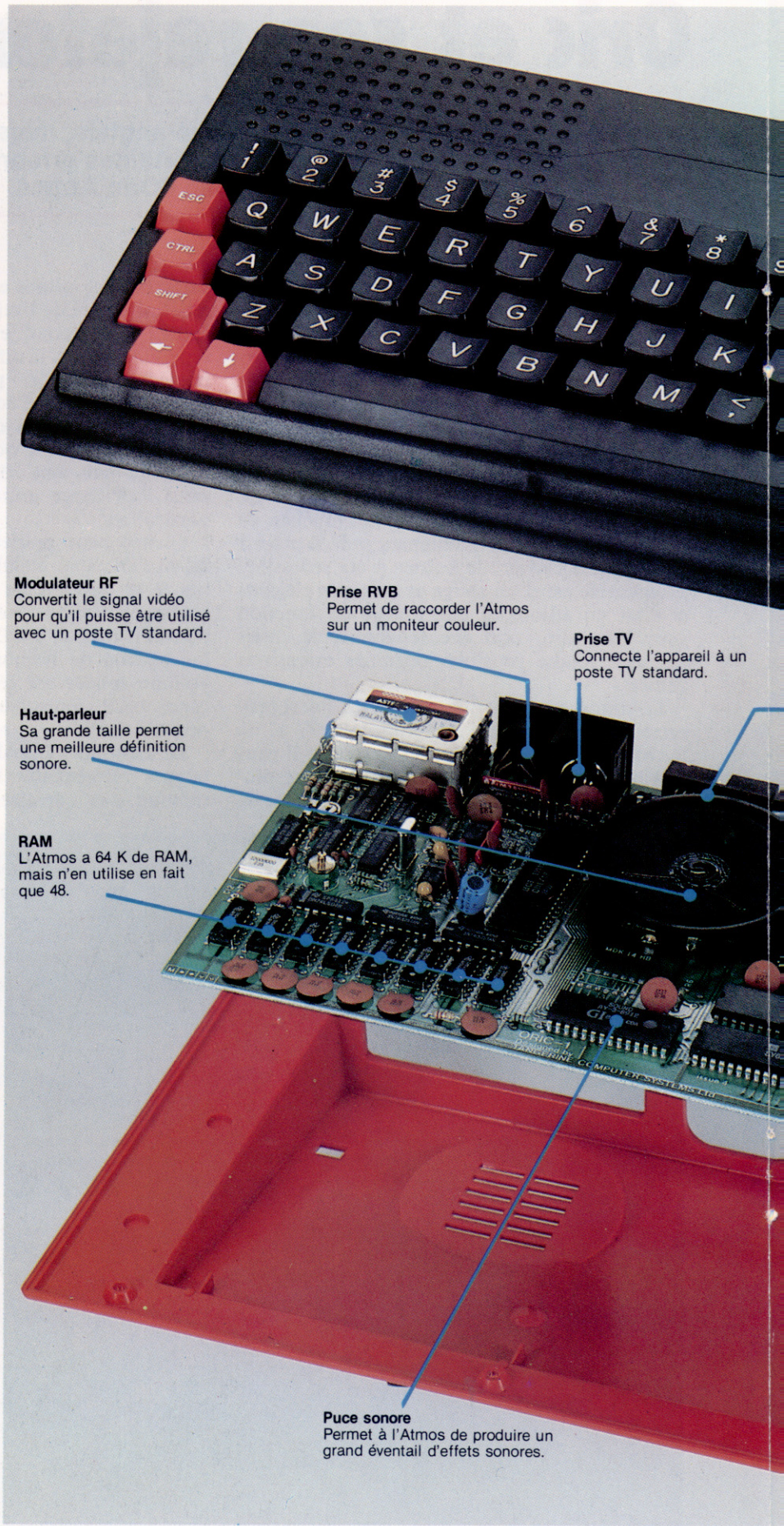
sonores perturbaient souvent l'affichage. La fonction STR$ était bizarrement contrôlée, et donnait des résultats incorrects en cas d'emploi de LEN ou de VAL. Tout cela a été rectifié lors de la création de l'Atmos; mais ces améliorations ont eu une conséquence ennuyeuse : le nouveau modèle ne pourra accepter les programmes en langage machine destinés à l'Oric-1, puisque certaines routines ROM ont été relogées ailleurs en mémoire.

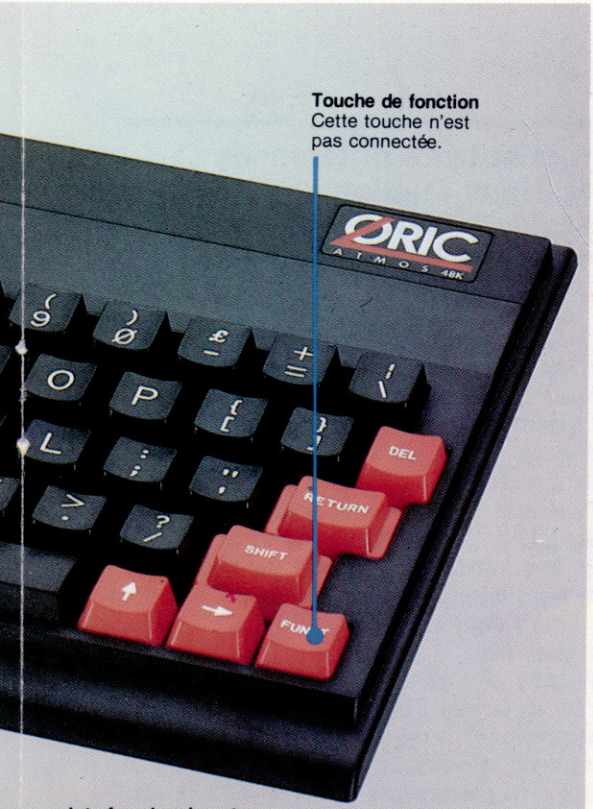
Mis au point par Tansoft à partir d'une version originale due à Tangerine, le BASIC est une variante étendue du dialecte Microsoft. Il dispose des structures IF...THEN...ELSE (ce dernier élément était défectueux sur l'Oric-1) et REPEAT...UNTIL. On peut, sans générer de message d'erreur, sortir de cette dernière, et aussi de GOSUB, grâce aux commandes particulières POP et PULL. Enfin il est désormais possible de POKEr des valeurs hexadécimales à telle ou telle adresse, contrairement à ce qui se passait avec le modèle précédent.

La création d'une nouvelle ROM a fait pourtant naître certains problèmes sur les premiers exemplaires de l'Atmos. Oric Products avait ainsi modifié la routine de recherche d'erreurs lors des chargements de programmes à partir d'une cassette. Le nouveau procédé était si efficace qu'il devenait à peu près impossible de charger quoi que ce soit. Ces difficultés ont heureusement été résolues depuis.

Oric Products a également redessiné son imprimante/table traçante, l'habillant des mêmes couleurs rouge et noir que l'ordinateur. Quatre petites pointes-bille (noire, rouge, verte et bleue dans la version de base) sont placées sur une tête traçante tournante; chaque teinte peut être sélectionnée par le programme. La vitesse d'impression est lente (12 caractères par seconde), mais un papier ordinaire suffit.

Le lecteur de disquettes, enfin disponible, est revêtu de la même livrée. Oric Products a finalement opté pour les disquettes trois pouces d'Hitachi, présentées dans un boîtier plastique

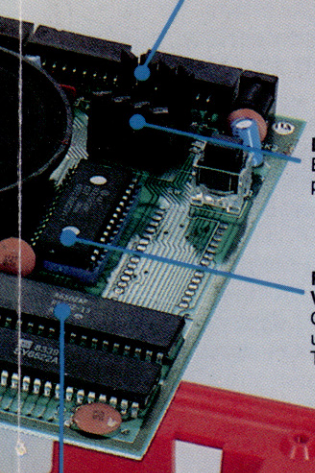




Touche de fonction
Cette touche n'est pas connectée.

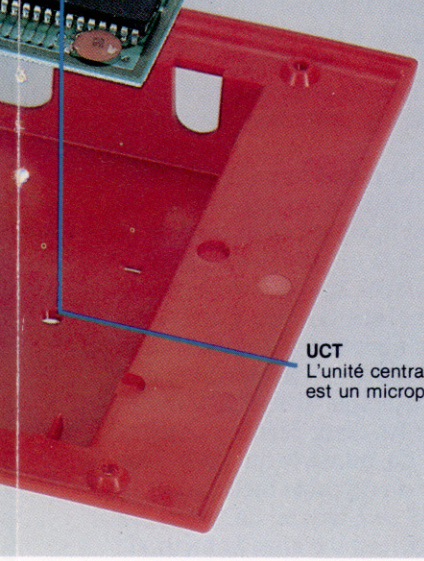
Interface imprimante
Interface parallèle de type Centronics.

Port d'extension
Contient un bus parallèle à 34 canaux, qui se raccorde au lecteur de disquettes.



Dissipateur de chaleur
Élimine la chaleur produite par les circuits électriques.

ROM BASIC
Version améliorée V1.1
Cette puce unique abrite une nouvelle version du BASIC Tansoft.



UCT
L'unité centrale de traitement est un microprocesseur 6502.

Chris Stevens

Lecteur de disquettes
L'appareil utilise des disquettes trois pouces. Elles sont vendues dans un boîtier rigide. Chaque face de disquette a une capacité de 160 K : on peut donc stocker 320 K en tout. Mais le lecteur ne peut traiter que des fichiers séquentiels, et non à accès direct, ce qui serait pourtant bien plus intéressant.

rigide. L'Atmos peut fonctionner avec un maximum de quatre lecteurs : une unité pilote (grâce à un système interface intégré) et trois unités asservies. Ces dernières n'ont pas encore été commercialisées, mais devraient l'être prochainement. Un transformateur fourni avec le lecteur est assez puissant pour faire fonctionner deux unités de disquettes ainsi que l'ordinateur lui-même. Là encore, les débuts n'allèrent pas sans mal : les premières versions du système d'exploitation faisaient naître des problèmes lorsque l'imprimante et le lecteur étaient branchés en même temps. Éditer une ligne de programme entraînait sa disparition du listage final, ainsi que du programme lui-même. Selon le constructeur, cette erreur a désormais été rectifiée.

En dépit de tous ces inconvénients, Oric Products semble avoir mûrement réfléchi à la conception de l'Atmos et de ses périphériques. Les concepteurs ont pris bonne note des critiques faites à l'Oric-1, et la plupart de ses erreurs ont été corrigées. Le premier modèle souffrait aussi d'un manque de logiciels; Tansoft a donc été chargé de rédiger des programmes utilisables avec le lecteur de disquettes.

Imprimante/Table traçante
Elle utilise quatre stylos à bille de couleur qui tracent du texte ou des lignes; les caractères peuvent aussi bien être en minuscules, ou faire plusieurs centimètres de haut. Inconvénients : faible largeur de papier, vitesse réduite, coût élevé des stylos à bille. Par ailleurs, l'appareil ne trace que des lignes, et ne peut donc remplir de vastes zones de couleur.

Chris Stevens

ORIC ATMOS

PRIX
**

DIMENSIONS
278 × 178 × 50 mm.

UC
6502.

MÉMOIRE
48 K de RAM, 16 K de ROM.

ÉCRAN
26 rangées de 40 caractères en mode texte et 200 × 240 pixels en haute résolution 8 couleurs.

INTERFACES
Un port d'extension pour une interface imprimante Centronics, un port cassette et une prise RVB.

LANGAGES DISPONIBLES
Basic étendu, FORTH.

CLAVIER
58 touches de type machine à écrire. La touche de fonction n'est pas connectée.

DOCUMENTATION
Le manuel est très complet; écrit dans un style très familier, il est avant tout destiné à faciliter la programmation en Basic. L'utilisateur averti tirera profit des chapitres consacrés au langage machine et aux entrées/sorties.
Les annexes techniques donnent des informations très détaillées.

FORCES
L'Atmos dispose de vastes possibilités dont des appareils plus coûteux sont dépourvus. Son Basic est concis, et ses nombreuses instructions facilitent la programmation.

FAIBLESSES
Il est assez difficile de travailler en haute résolution en raison de l'affichage écran.
Les disquettes ne sont accessibles qu'en mode séquentiel et perdent donc beaucoup de leur intérêt.

Positions clés

Nous avons déjà vu les fichiers à accès direct et leurs différences fondamentales avec les fichiers séquentiels. Nous étudions aujourd'hui leur organisation avec index et table mêlée.

Si vous avez déjà utilisé des fichiers à accès direct, vous savez qu'ils facilitent la programmation. Vous pouvez spécifier un enregistrement (en lecture ou en écriture) sans vous soucier des lourdes procédures d'accès aux données stockées séquentiellement. Pourtant, les méthodes d'insertion et de suppression d'enregistrements que nous avons vues précédemment ne sont pas les plus efficaces. En effet l'utilisation d'un index pour accéder à l'information se révèle très efficace.

Pour créer un index, il faut attribuer une clé à chaque zone des enregistrements. La valeur de cet index permettra ensuite d'accéder aux enregistrements recherchés pour affichage ou traitement. L'index est constitué de la valeur de la zone clé pour chaque enregistrement, avec le numéro d'enregistrement correspondant. Par exemple, si l'enregistrement pour Benoît Gourd a le numéro d'enregistrement 17, et s'il correspond au 8e enregistrement de la liste alphabétique, la table d'index stockera la valeur 17 en 8e position. Si l'index est régulièrement trié, la recherche d'un enregistrement sera très rapide.

L'index est généralement stocké en RAM afin d'être plus rapidement accessible. Une routine spéciale permet de le créer instantanément par l'intermédiaire d'une routine qui lit la totalité du fichier, en plaçant au fur et à mesure la zone clé dans un tableau. Cet index est ensuite trié pour être prêt à l'emploi. Cela est assez long. Une autre méthode consiste à stocker les fichiers d'index sur disque comme les fichiers de données. Il est ainsi possible de créer autant de zones clé qu'on le désire, par l'intermédiaire de fichiers d'index sur disque. Cela permet au fichier d'être indexé de différentes façons : les enregistrements peuvent être restitués selon le nom (de A à Z ou de Z à A), la date, ou selon d'autres critères encore.

Comment stocker un fichier d'index? Il doit comporter deux zones, les données correspondant à la clé et le numéro de l'enregistrement. Ces zones seront chargées en mémoire pour y être utilisées et elles ne seront réécrites au fichier que pour modification ou mise à jour. Ce type d'application convient parfaitement à une organisation séquentielle comme nous l'avons vu par opposition à l'accès direct : les données doivent être transmises dans l'ordre où elles sont stockées. Nous avons donc un exemple où les deux types d'organisation se complètent.

Supprimer un enregistrement d'un fichier à accès direct indexé revient à l'indiquer comme devant être détruit et à s'assurer qu'il ne figure plus à l'index. La manière la plus simple de le faire consiste à placer un marqueur « enregistrement supprimé » dans l'enregistrement — un astérisque au début de la première zone par exemple. En effet, la totalité de la zone remplie d'un drapeau « destruction » serait une perte de place. La clé de l'enregistrement ainsi marqué pourrait être retirée de l'index, ou encore, le numéro de l'enregistrement pourrait prendre une certaine valeur signifiant qu'il a été détruit, – 1 par exemple.

Quelle que soit la méthode retenue, il faut qu'il y ait au fichier l'indication des enregistrements supprimés. Lorsque de nouveaux enregistrements sont ajoutés, ils peuvent prendre la place des enregistrements à détruire. Les clés originelles d'entrée à la table, pour ces enregistrements réécrits, doivent être remplacées par de nouvelles. Le fichier d'index devra ensuite être trié à nouveau afin de classer les nouveaux enregistrements. Le programme permettra de la sorte de pouvoir recouvrer les enregistrements effacés par erreur, pourvu qu'ils n'aient pas été réécrits entre-temps.

Il est important de prévoir la remise en ordre du fichier dans la mesure où notre système d'indexation stocke les nouveaux enregistrements dans le désordre et avec de nombreux trous.

L'indexation n'est pas le seul moyen de trouver rapidement un enregistrement dans un très grand fichier. La technique dite de *table mêlée* s'applique à de très gros fichiers et se rencontre surtout sur des systèmes à disques durs, ou sur des systèmes de disquettes à très grande capacité mémoire. Cependant, de nombreux systèmes l'utilisent de manière interne pour accélérer leur vitesse d'exécution. C'est donc une technique qu'il faut connaître.

La technique de table mêlée remplace l'index par une formule arithmétique ou algorithme de table mêlée. Elle prend la valeur de la zone de la clé et génère à partir de là un numéro d'enregistrement. L'enregistrement qui accompagne la clé est stocké sur cette position au fichier. La formule dépend de la nature des données de la zone. S'il s'agit d'une date — trichronologique —, vous pourrez par exemple utiliser le numéro du mois multiplié par les deux derniers chiffres de l'année et ajouté du numéro du jour. Une zone-nom sera traitée par la table mêlée en utilisant les codes ASCII des lettres du nom. La nature de la zone clé détermine ainsi la méthode

d'élaboration de l'algorithme de la table. Supposons que nous voulions créer un fichier de table mêlée avec les enregistrements des employés d'une société, sur la clé de leurs prénoms. L'algorithme sera : les codes ASCII des quatre premières lettres sous la forme d'un nombre à 8 chiffres, mis au carré pour garder finalement les quatre derniers chiffres du nombre. De la sorte les données pourront être séparées. JONES se place ainsi dans l'enregistrement 1161, et JONQUIL dans l'enregistrement 0161.

Le hachage est moins souple mais plus rapide que l'indexation. Pour localiser un enregistrement donné, le programme utilise sa clé, calcule sa position par partition des données et restitue le contenu de l'enregistrement. Le temps gagné est celui de la recherche dans une table (sans parler de la création de la table).

Il se peut que deux enregistrements génèrent le même code haché et entrent en conflit car ils devraient occuper la même position au fichier. Pour résoudre ce problème, les algorithmes doivent être soigneusement étudiés de sorte que deux clés différentes, et à plus forte raison identiques, ne puissent produire le même hachage.

Nous pouvons maintenant résumer une table mêlée (hachage) : lorsqu'un enregistrement est stocké, sa clé est soumise à l'algorithme de partition des données, pour générer un numéro d'enregistrement. Si le numéro produit est déjà attribué, le système passe à l'enregistrement suivant selon l'ordre séquentiel. Cette opération peut se renouveler sur tout le bloc d'enregistrements (5 enregistrements généralement), associé à ce code de partition. Pour lire un enregistrement, sa clé est hachée et le bloc correspondant est passé en revue séquentiellement pour trouver l'enregistrement. On peut penser que cela revient à annuler tout le bénéfice de la vitesse de l'algorithme, mais le but du hachage est seulement de réduire le nombre d'enregistrements à passer en revue, dans un rapport de 3000 à 5 ou 6.

Que se passe-t-il lorsque tous les enregistrements d'un bloc sont occupés? La solution la plus évidente est de transmettre un message « fichier plein ». Le plus souvent, les enregistrements ne pouvant figurer dans la table mêlée sont inscrits dans un fichier supplémentaire qui intervient lorsqu'il y a dépassement de la capacité mémoire. Ce fichier comprend son propre index et est susceptible d'être réintégré au fichier principal. Pourtant, la plupart des systèmes évitent ce genre d'artifice en ne remplissant qu'à 80 % ou même moins, ces fichiers.

L'organisation à table mêlée accélère par ailleurs la suppression d'enregistrements. Il suffit de hacher la clé de l'enregistrement, de le localiser dans le bloc, et d'indiquer sur sa position qu'il est vide. Il pourra alors être recouvert en écriture au prochain enregistrement ayant le même code de partition.

Dans le dernier article de cette série, nous étudierons les commandes BASIC de création et d'accès des fichiers sur cassettes.

Fichier indexé

Le mode d'accès le plus courant pour un fichier direct est l'indexation. Il y a alors une liste en RAM des valeurs de la zone clé pour les différents enregistrements. Pour accéder à un enregistrement, il suffit de rechercher sa clé dans l'index et de la charger en mémoire.

Les enregistrements à supprimer sont laissés au fichier et simplement marqués comme tels. Ils peuvent alors être remplacés par de nouveaux enregistrements.

Trouver « David »

Clé	N°
André	1
Benoît	−1
Bernard	5
Cristèle	−1
David	7
Dawes	23
Fernand	15
Grégoire	28
Henri	37
Jean	25
Klaus	11
Marc	10

Fichier indexé

Fichier principal

	Nom	Tél. Bur.	Tél. Dom.	Qualification
1	André	142 0791	127 0942	Architecte
2	Philippe	136 2418	121 3940	Comptable
3	Smith	131 0836	186 8170	Rédacteur
4		Enregistrement supprimé		
5	Brown	129 8213	136 2190	Dentiste
6	Pierre	136 6622	198 4310	Décorateur
7	David	143 7316	150 6926	Jardinier
8		Enregistrement supprimé		
9		Enregistrement supprimé		
10	Marc	130 6321	129 7592	Mécanicien
11	Klaus	193 9899	155 8431	Juriste
12	Watteau	136 7700	193 0452	Coiffeur

Table mêlée

ALGORITHME DE PARTITION

Cet algorithme traduit la clé en un code qui renvoie à un bloc déterminé d'enregistrements. Les enregistrements de même code de partition sont groupés en un même bloc.

Les espaces laissés entre les blocs servent de tampons pour recevoir les nouveaux enregistrements apparentés ainsi insérés en bon ordre.

Nom	Tél. Bur.	Tél. Dom.	Qualification
Dazibao	129 0491	130 0592	Plombier
Davout	136 2488	162 0066	Administrateur
Daran	130 0021	126 9191	Teinturier
Dammat	139 9933	130 4918	Écrivain
David	143 7216	150 6926	Jardinier
Dawes	130 0123	140 9924	Puéricultrice
Egerton	131 6666	158 0021	Architecte
Erarh	131 8294	150 6218	Grossiste

Les fichiers à table mêlée permettent un accès très rapide aux enregistrements recherchés dans des fichiers directs de grande taille, mais ils manquent de souplesse.

La clé d'enregistrement est soumise à un algorithme qui la transforme en un code de partition permettant de situer l'enregistrement dans le fichier. Ces codes correspondent le plus souvent à des blocs d'enregistrements.

Kevin Jones

Jeux à entendre

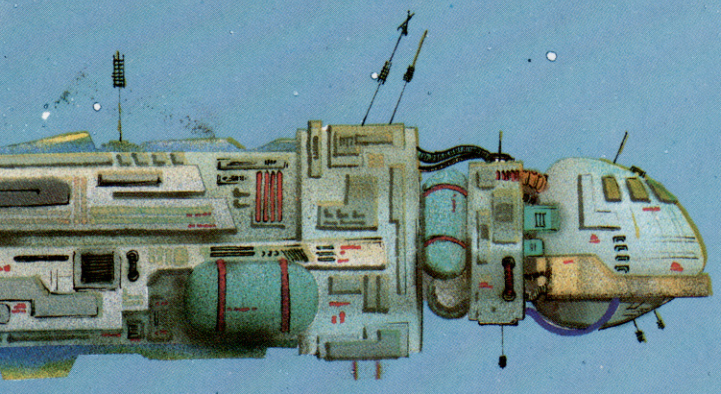
```
   10 REM******JEU SONORE******
***
   20 PRINT" UN SIMPLE JEU SON
ORE":PRINT:PRINT" DIRIGEZ VOTR
E VAISSEAU VERS LA BALISE AVAN
T D'ETRE A COURS DE CARBURANT
   30 PRINT:PRINT" PLUS VOUS A
PPROCHEZ DE LA BALISE PLUS LE
SIGNAL SONORE EST ELEVE
   40 PRINT:PRINT" VOS COMMAND
ED SONT:":PRINT I (vers le hau
t, M (vers le bas), J (à sauch
e), K (à droite)"
   60 PRINT:PRINT "******* frap
pez une touche pour commencer
*******"
   70 A$=INKEY$(0):IF A$="" TH
EN GOTO 70
   80 P=INT(RND(1)*500)+1:Q=IN
T(RND(1)*500+1):F=3*(P+Q)/2:X=
0:Y=0
   90 XD=0:YD=0
  100 PRINT "CARBURANT=";F, "
DISTANCE=";
  110 IF (ABS(P-X) 2 AND ABS(Q
-Y)<2) THEN PRINT"YOU MADE IT
- WITH ";F;" FUEL UNITS LEFT":
END
  120 A$=INKEY$(0):IF A$="" TH
EN GOTO 120
  130 IF (A$="I" AND YD 3) THE
N YD=YD+1
  140 IF (A$="M" AND YD>-3) TH
EN YD=YD-1
  150 IF (A$="J" AND XD>-3) TH
EN XD=XD-1
  160 IF (A$="K" AND XD<3) THE
N XD=XD+1
  170 X=X+XD:Y=Y+YD
  180 D=SQR((P-X)*(P-X)+(Q-Y)*
(Q-Y))
  190 PRINT D
  200 SOUND 1,-8,(255-D),2
  210 F=F-ABS(XD)-ABS(YD):IF F
>0 THEN GOTO 90
  230 PRINT TAB(10);"*******CR
ASH*******":PRINT TAB(11);"***
PANNE DE CARBURANT***"
  240 PRINT:PRINT" VOTRE POSIT
ION EST";D; UNITES DE DISTANC
E DE L'ESPACE DE VOTRE BASE
  260 END
```

Les jeux sur ordinateur les plus célèbres font un très large usage des effets visuels impressionnants que peut produire la micro-informatique. Mais si de nombreux jeux sont très bruyants, peu sont réellement créatifs pour le son.

Les jeux d'arcade utilisent le son pour compléter leurs effets graphiques. Bien que les effets sonores servent à accroître le réalisme et l'attrait du jeu, ils sont rarement en rapport direct avec lui. Et pourtant, si nous utilisons notre vue comme principal « récepteur » des jeux sur ordinateurs, il n'y a pas de raison de ne pas en faire autant pour notre ouïe.

Le jeu que nous esquissons ici est en quelque sorte un « jeu sonore ». Nous ne prétendons pas offrir le jeu le plus passionnant qui ait été inventé, mais c'est certainement un jeu captivant. Ce qui était au départ un simple travail d'informaticien devient d'un grand intérêt car le jeu fait appel à des sensations différentes de celles exprimées lors du contrôle d'une image bariolée de couleurs vives.

Dans notre jeu, vous vous trouvez aux commandes d'un vaisseau spatial qui va manquer de carburant. Votre seul espoir de survie n'est plus, désormais, que de pouvoir accoster à une station orbitale proche où vous pourrez vous ravitailler. Cet espoir est réduit par une panne de vos instruments de bord due à un mauvais fonctionnement de votre ordinateur de route. Vous avez perdu tout contrôle visuel. La seule méthode qui vous reste est le signal sonore émis par sa balise. L'intensité du signal s'accroît au fur et à mesure que vous vous en approchez. Il vous faudra donc être attentif au niveau sonore et réagir sur les commandes de votre vaisseau en conséquence.



Dave Cooper/Smith

Tout comme un vaisseau spatial réel, votre engin suivra sa trajectoire jusqu'à modification par des forces de réaction. Si vous utilisez les forces « Deux Us » pour avancer plus vite, il vous faudra leur opposer les forces « Deux Ds » pour vous arrêter. Cela rend le jeu plus complexe qu'il n'y paraît.

Il vous faudra de l'entraînement pour parvenir à diriger votre vaisseau uniquement selon les indices auditifs. En ajoutant quelques lignes au programme, vous pourrez, il est vrai, obtenir sur votre écran une indication de la position du vaisseau par rapport à la station. Cela pourrait être, par exemple, l'indication de la distance (la variable D), ou encore celle des commandes à utiliser de préférence (si SGN(p-x)=-1, alors il vous faudra aller sur la gauche, par exemple).

Ayant mis en pratique votre programme, vous voudrez peut-être créer vos propres jeux. L'utilisation du son pour repérer ou éviter des objets est un domaine particulièrement riche et inexploité. Pourquoi ne pas créer des jeux avec sonar pour sous-marins, ou encore de navigation à travers un champ de mines avec un détecteur qui émet un faible signal d'avertissement? Cela doit être relativement facile.

Variantes de basic

Ce programme a été écrit pour un BBC Micro en Mode 7; il est donc pratiquement au standard BASIC Microsoft. Ses commandes PRINT supposent l'affichage sur 40 colonnes et le formatage pour plusieurs modes d'affichage. La commande SOUND à la ligne 200 est spécifique au Basic BBC. La valeur du paramètre (255-D) représente l'intensité sonore de la note à jouer, les autres paramètres contrôlent le volume, la durée et le canal pour le son. INKEY$(0) à la ligne 120, et RND à la ligne 80 seront à réécrire.

Spectrum

Mettre LET dans toutes les instructions d'assignation.
Remplacez INKEY$(0) par INKEY$, et RND(1) par RND.

```
200 BEEP 0.4,(255-D)
```

Ajoutez les lignes suivantes :

```
15 RANDOMIZE
195 IF D>254 THEN LET D=D-254
```

Commodore 64/Vic-20

Remplacez INKEY$(0) par GET A$. Voyez votre manuel utilisateur pour les commandes son. Ajoutez :

```
75 X=RND(-TI)
```

Dragon

Remplacez INKEY$(0) par INKEY$. Remplacez RND(1) par RND(0).

```
200 SOUND (255-D),10
```

Ajoutez la ligne :

```
195 IF D>254 THEN D=D-254
```

Oric Atmos

Remplacez INKEY$(0) par KEY$. Modifiez ainsi la ligne 200 :

```
200 SOUND 1,(255-D),9:WAIT 40:PLAY 0,0,0,0
```

et ajoutez :

```
195 IF D>254 THEN D=D-254
```

Sport aquatique

Le marché du logiciel étant actuellement dominé par les jeux de type « guerre de l'espace », il est agréable de découvrir un nouveau jeu qui sort des sentiers battus et nous fait connaître un tout autre horizon.

Scuba Drive est disponible en trois versions : l'une destinée au Spectrum écrite par Mike Richardson, une autre pour l'Oric-1 créée par Ron Jeffs et une dernière écrite par Nigel Dewdney pour le Commodore 64. La version de l'Oric-1 a été modifiée afin de pouvoir être également exécutée sur l'Oric Atmos.

Le joueur tient le rôle d'un plongeur qui risque sa vie en recherchant un trésor sur le fond marin. Le principal objectif de notre héros est de rassembler un nombre maximum de perles qu'il trouve dans des coquilles d'huître ou dans des palourdes géantes. A un niveau plus évolué du jeu, votre recherche s'effectue dans des cavernes sous-marines. De nombreuses bonnes raisons incitent à une grande prudence pendant cette recherche. L'eau est peuplée de créatures qui représentent un danger de tous les instants : méduses, pieuvres, anguilles électriques, et requins dans la version du Spectrum! Vous devez les éviter à tout prix. Vous découvrirez un autre danger lorsque vous commencerez à extraire des perles de l'intérieur des palourdes géantes. Ces créatures peuvent se refermer sur vous et vous emprisonner.

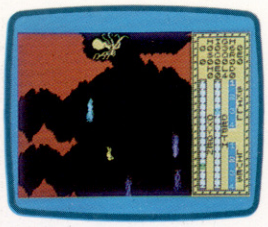
Les entrées étroites de la caverne principale et des galeries inférieures sont gardées par des pieuvres géantes. Difficile d'échapper à leurs tentacules... Mais il existe, quand même, toujours un moyen pour les contourner. La version du Commodore ne comporte pas de pieuvres, mais une porte piège. Celle-ci s'ouvre et se ferme constamment pendant que vous essayez de la traverser sans qu'elle ne vous heurte.

Le programme vous attribue trois vies par jeu et offre un niveau de difficulté allant de 1 à 4. Une vie est perdue lorsque vous touchez à l'un des membres de la faune aquatique ou lorsque vous avez épuisé votre oxygène. Chaque version affiche d'abord une vue de la surface de la mer et aussi une grande partie des profondeurs. Le bateau d'où vous plongez tangue à la surface. Vous risquez de rester coincé sous le navire au moment où vous plongez; vous devez donc être très prudent lors de la descente. Sur la version du Spectrum, il est préférable d'avoir un bon sens de l'orientation, puisque votre bateau peut dériver pendant que vous êtes sous l'eau. Dans la version de l'Oric, ce n'est pas un problème, puisque le bateau se déplace toujours de gauche à droite et qu'il n'est jamais hors de vue lorsque vous faites surface.

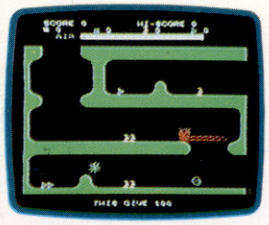
La qualité graphique de la version Spectrum est superbe. La couleur a très bien été utilisée et les créatures sous-marines semblent assez réelles. Les déplacements du plongeur sont commandés à l'aide des touches X et Z qui le font tourner dans le sens des aiguilles d'une montre ou en sens inverse. Les touches Espace et Shift font avancer le plongeur. Les propriétaires de Spectrum peuvent aussi utiliser des manches à balai, bien que le programme ne fonctionne pas avec l'interface de manche à balai Kempston. Les possesseurs d'un Commodore ont eux aussi la possibilité d'utiliser des manches à balai. Sur l'Oric, les joueurs doivent se contenter du clavier.

La version Oric est à plusieurs points de vue beaucoup moins captivante. Le mouvement du plongeur et des créatures est très flou, et les graphiques — surtout les murs des cavernes — comportent beaucoup moins de détails. Pour sa part la version Commodore offre elle aussi moins de détails que celle du Spectrum, mais elle reste nettement meilleure que celle offerte sur l'ordinateur Oric.

Comparaison de la qualité
Ces images d'écran illustrent la différence de qualité entre les versions du jeu Scuba Dive.

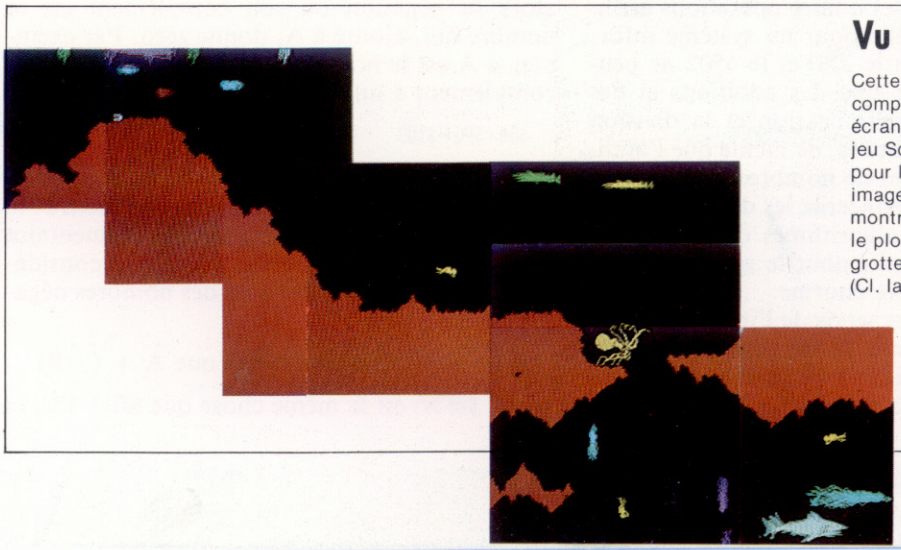


Sinclair Spectrum



Oric-1

Commodore 64



Vu en perspective

Cette image est composée de plusieurs écrans de la version du jeu Scuba Dive créée pour le Spectrum. Les images assemblées montrent comment le plongeur voit les grottes sous-marines. (Cl. Ian McKinnell.)

Adresse enregistrée

Nous allons commencer à regarder de plus près l'exécution de procédures simples en langage machine, comme les opérations arithmétiques fondamentales que sont l'addition et la soustraction.

Les différences de fonctionnement entre les microprocesseurs Z80 et 6502 révèlent des philosophies de conception particulière. Les nombreux registres du Z80, avec leur jeu sophistiqué d'instructions, sont typiques du processeur lui-même — élégant, complexe et puissant. L'architecture et le jeu d'opérations bien plus simples du 6502 semblent suggérer un processeur à la fois plus modeste, robuste et pratique. Cette impression est exacte dans une certaine mesure, mais l'abondance de modes d'adressage du 6502 et son utilisation de la page zéro en registre d'index supplémentaire lui confèrent une subtilité et une souplesse qui lui permettront de dominer le monde de la micro familiale et professionnelle encore pendant quelque temps.

Le grand avantage des registres du Z80 est leur flexibilité — ils peuvent être traités simultanément à la fois comme registres à un octet ou à deux octets, ce qui autorise un très large éventail d'adressage. Par ailleurs, le 6502 n'a pas de registres à deux octets, mais il est capable de traiter la page zéro comme une série de registres à un ou deux octets.

Principes arithmétiques

Nous avons vu que les registres d'UC permettent une variété d'accès possibles à la mémoire; mais pour manipuler celle-ci, il faut généralement quelque chose de plus que simplement charger, stocker et comparer son contenu. La capacité d'effectuer les quatre opérations arithmétiques est essentielle pour un système informatique, et pourtant le Z80 et le 6502 ne peuvent tous deux faire que des additions et des soustractions. La multiplication et la division doivent être programmées, de même que l'addition et la soustraction de nombres supérieurs à $FF. Cette limitation concerne les deux UC, bien que l'invention des algorithmes de multiplication et de division soit, pour le programmeur, d'une valeur éducative énorme.

Nous nous sommes servis de l'instruction ADC (addition avec retenue) et de diverses instructions INC (incrémentation), en faisant de l'arithmétique à un octet sur les deux UC. Voici les deux façons d'additionner les contenus de deux positions mémoire à deux octets :

6502			Z80		
ADDR1	DW	$7E60	ADDR1	DW	$7E60
ADDR2	DW	$4A51	ADDR2	DW	$4A51
SUM	DS	$03	SUM	DS	$03
BEGIN	CLC		BEGIN	LD	A,$00
	LDA	ADDR1		AND	A
	ADC	ADDR2		LD	HL,(ADDR1)
	STA	SUM		LD	DE,(ADDR2)
	LDA	ADDR1+1		ADD	HL,DE
	ADC	ADDR2+1		LD	(SUM),HL
	STA	SUM+1		ADC	A,$00
	LDA	$00		LD	(SUM+2),A
	ADC	$00		RET	
	STA	SUM+2			
	RTS				

La méthode à un octet employée sur le 6502 peut être utilisée sur Z80, mais la méthode à deux registres utilisée dans la version Z80 n'a pas d'équivalent en 6502. Notez les stratégies utilisées pour manier les diverses possibilités de retenue, depuis les instructions CLC (6502) et AND A (Z80) qui mettent à zéro le drapeau de retenue avant l'addition, jusqu'à la modification du troisième octet de SUM. Autoriser le plus grand résultat possible est vital dans toute l'arithmétique.

La soustraction peut être traitée de manière analogue à l'addition, les deux processeurs possédant une instruction SBC (''substract with carry'' : soustraire avec retenue), quoique le Z80 puisse accomplir des soustractions sur deux octets. La possibilité d'avoir des résultats négatifs en soustraction nous entraîne à examiner la représentation binaire du signe algébrique.

Pour commencer, contentons-nous de dire que :

si A + B = 0 alors il s'ensuit que A = −B

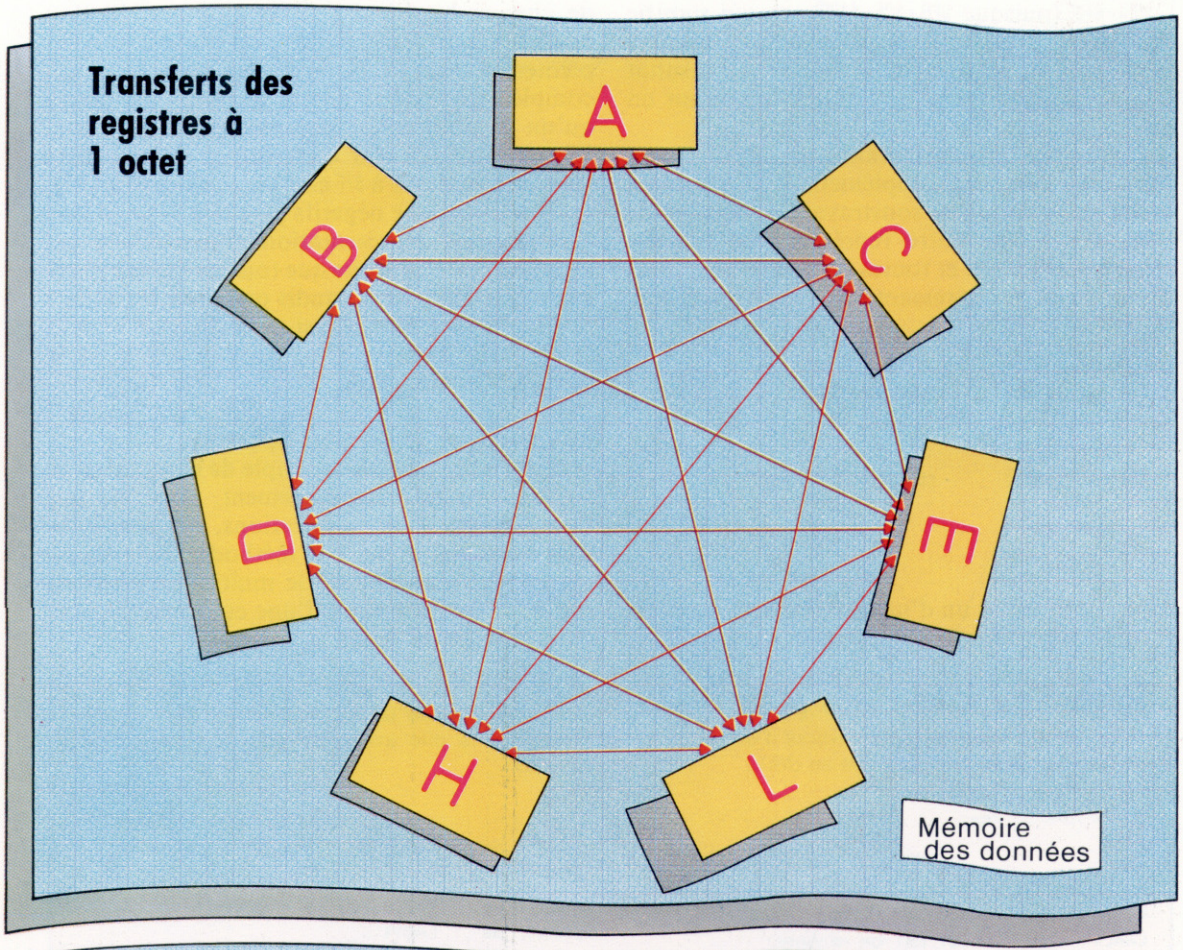
ce qui implique que, si A est un nombre positif, alors sa négation ou son complément est le nombre qui, ajouté à A, donne zéro. Par exemple, si A est le nombre à un octet $04, alors son complément à un octet est $FC :

$04+$FC=$100

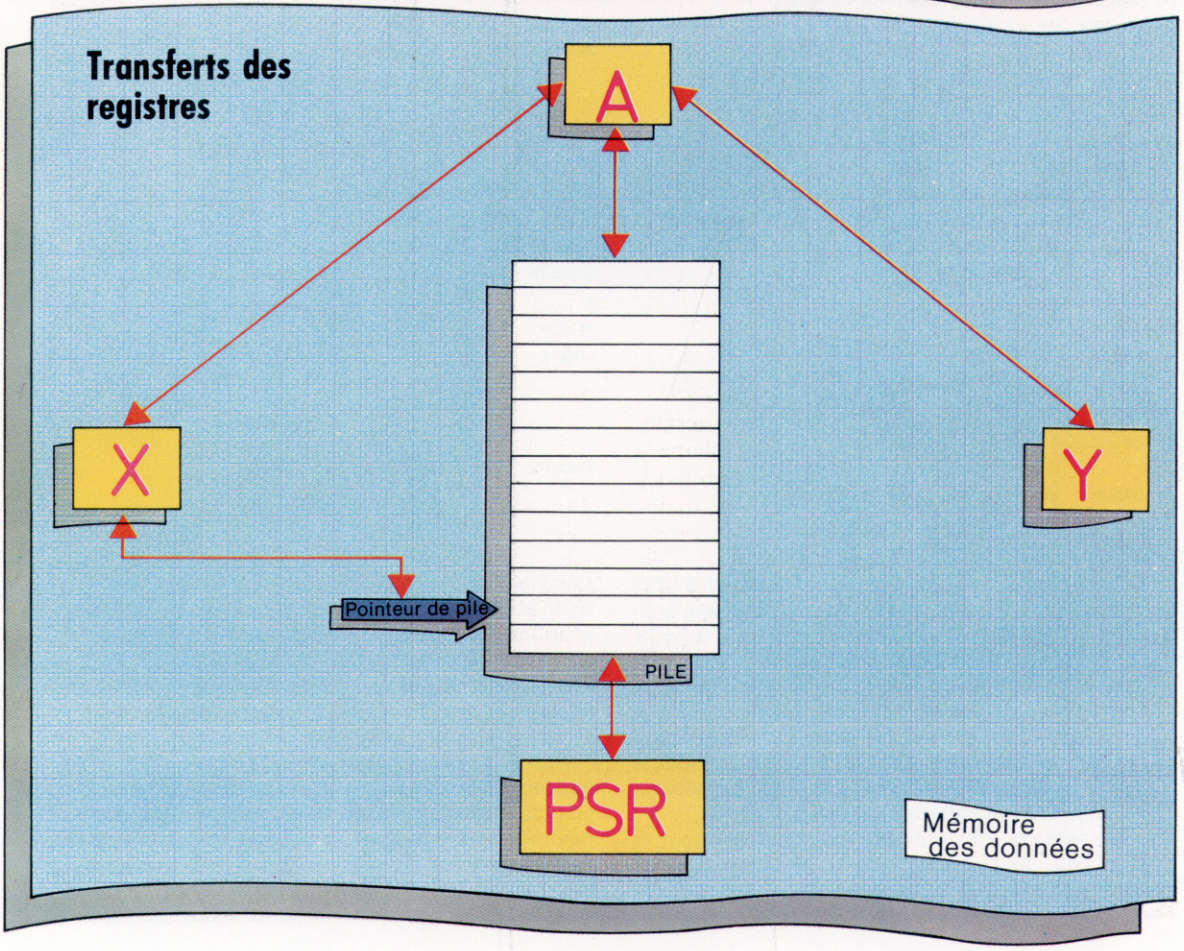
En nous rappelant que $100=$00 (si nous n'avons qu'un registre à un octet pour mettre le résultat), cette représentation complémentaire implique que la soustraction peut être considérée comme une addition avec des nombres négatifs. C'est-à-dire :

A − B est la même chose que A + (−B)

Ainsi, $08-$05 est la même chose que $08+(−$05), et



Double identité
Les registres de données du Z80 peuvent communiquer comme des registres à un octet avec tout autre registre à un octet. La communication avec la mémoire peut être en mode direct, immédiat, indirect, absolu et indexé. Lorsqu'ils sont traités comme BC, DE, HL — les paires de registres à deux octets — ils peuvent transférer des données de 16 bits entre la mémoire et la pile; ce sont effectivement des accumulateurs 16 bits pour l'addition et la soustraction. Cette combinaison de souplesse et de richesse fait le succès du Z80. (Cl. Kevin Jones.)



Pur et simple
La communication interne du 6502 est strictement linéaire, et restreinte aux transferts de données sur huit bits. Seul l'accumulateur peut communiquer directement avec X et Y; seul X peut communiquer avec le pointeur de pile; et seuls le PSR et l'accumulateur ont accès à la pile. Les transferts de mémoire sont possibles dans les modes absolu, direct, indirect, indexé, immédiat et page zéro. L'utilisation originale du mode page zéro par le 6502 compense la petite taille de son ensemble de registres; la page zéro peut être traitée comme 128 registres d'UC à deux octets. (Cl. Kevin Jones).

(-$05)=$FB (puisque $FB+$05=$100), ce qui signifie que notre problème de soustraction originel peut être reformulé comme $08+$FB. Le résultat de cette somme est $103, ce qui donne $03 sur un octet.

Cette sorte de représentation est appelée *complément à deux* : le complément d'un nombre à un octet est formé en soustrayant ce nombre de $100. Il y a une autre représentation appelée *complément à un,* et toutes les deux sont reliées d'une manière intéressante. Considérons l'opération suivante :

```
   $05 = 00000101        binaire
$FA    = 11111010        complément à un
             +1
   ------------------------
   $FB = 11111011        complément à deux
   ------------------------
   $05+$FA=$FF
   $05+$FB=$00
```

Le complément à un d'un nombre à un octet est formé en prenant le complément ou le négatif de chaque bit binaire de ce nombre. Si l'on additionne un à ce résultat, on obtient le complément à deux du nombre. Un nombre et son complément à un totalisent toujours $FF, tandis qu'un nombre et son complément à deux font $00 (en fait $100). Par convention, on considère les nombres de $00 à $7F comme positifs (0 à 127), et $80 à $FF comme négatifs (– 128 à – 1). Si l'on compare les représentations binaires de ces nombres, on remarque que tous les entiers négatifs ont le bit 7 à un, tandis que pour les entiers positifs il est à zéro. C'est pourquoi, on appelle le bit 7 *bit de signe* d'un nombre relatif, et le drapeau de retenue du registre d'état du processeur (PSR) est mis ou non, selon l'état du bit 7 du résultat de la dernière opération.

Il n'y a pas de moyen simple de contourner ce sujet déroutant. Heureusement, une fois que l'on a compris ses implications, on peut l'utiliser machinalement. Ces méthodes empiriques, comme les algorithmes de multiplication et de division, feront l'objet d'une étude particulière dans la suite du cours.

Solutions des exercices précédents

1. Le programme suivant inverse l'ordre de la chaîne de caractères stockée en LABL1 :

```
                6502
;
ORIGIN  ORG   $7000
LAST1   EQU   $0D
LABL1   DB    'CECI EST UN MESSAGE'
TERMN8  DB    LAST1
;
BEGIN   LDX   #$FF
        LDA   #LAST1
        PHA
LOOP0   INX
        LDA   LABL1,X
        PHA
        CMP   #LAST1
ENDLP0  BNE   LOOP0
CLRSTK  PLA
;
BEGIN1  LDX   #$FF
LOOP1   INX
        PLA
        STA   LABL1,X
        CMP   #LAST1
ENDLP1  BNE   LOOP1
        RTS
```

Dans la version 6502, le code compris entre LOOP0 et ENDLP0 utilise l'adressage indexé X dans une boucle pour charger les caractères un par un à partir de LABL1, et les entre sur la pile — après avoir entré la valeur ASCII du caractère terminal marquant le bas de la pile. Le dernier caractère entré sur la pile est aussi le terminateur, déterminé par sa position de dernier caractère dans la chaîne. La boucle s'achève ainsi, et le caractère terminal sur le haut de la pile est ensuite remis à zéro en CLRSTK.

La version Z80 utilise l'adressage indirect IX pour charger l'accumulateur à partir de LABL1, et entre non seulement l'accumulateur mais aussi le registre de drapeau, sur la pile. Cela signifie que les caractères de la chaîne en LABL1 sont disséminés sur la pile avec des valeurs successives du PSR.

```
                Z80
        ORG   $C000
LAST1   EQU   $0D
LABL1   DB    'CECI EST UN MESSAGE'
TERMN8  DB    LAST1
;
BEGIN   LD    IX,LABL1-1
        LD    A,LAST1
        PUSH  AF
LOOP0   INC   IX
        LD    A,(IX+0)
        PUSH  AF
        CP    LAST1
ENDLP0  JR    NZ,LOOP0
CLRSTK  POP   AF
;
BEGIN1  LD    IX,LABL1-1
LOOP1   INC   IX
        POP   AF
        LD    (IX+0),A
        CP    LAST1
ENDLP1  JR    NZ,LOOP1
        RET
```

Dans les deux versions, le code compris entre BEGIN1 et ENDLP1 est le reflet de la boucle précédente et utilise les mêmes techniques, mais cette fois en sortant les caractères de la pile dans l'ordre inverse et en continuant à les stocker en LABL1. La boucle se termine lorsque le caractère terminal est trouvé au bas de la pile.

Notez l'importance de l'équilibre entre les entrées et sorties de la pile; la partie la plus difficile du problème consiste à décider comment manipuler les conditions extrêmes — que faire au début des boucles, comment les terminer et comment organiser tout cela.

L'instruction en BEGIN et BEGIN1 (LD IX, LABL1-1) illustre l'utilité d'un programme assembleur. Ici, il décode l'expression (LABL1-1) comme « l'adresse de l'octet précédant immédiatement l'octet dont l'adresse est LABL1 », et assemble cette adresse en code. La plupart des assembleurs comportent, plus ou moins, une évaluation d'expression permettant généralement de modifier un ou deux opérandes par un seul opérateur arithmétique — normalement « + » ou « – ».

2. Ce programme inverse l'ordre des caractères dans chaque mot de la chaîne en LABL1, tout en conservant l'ordre des mots :

```
                6502
;
ORIGIN  ORG   $7000
LAST1   EQU   $0D
SPACE   EQU   $20
LABL1   DB    'CECI EST UN MESSAGE'
TERMN8  DB    LAST1
;
BEGIN   LDX   #$FF
LOOP0   JSR   RVSWRD
        CMP   #LAST1
ENDLP0  BNE   LOOP0
        RTS
;
;****SP  INV.  D'UN MOT ****
LASTCH  DB    $00
LASTX   DB    $00
RVSWRD  TXA
        TAY
        INY
RVSLP0  INX
        LDA   LABL1,X
        PHA
        CMP   #SPACE
        BEQ   CLRSTK
        CMP   #LAST1
ENDRV0  BNE   RVSLP0
CLRSTK  PLA
        STA   LASTCH
        STX   LASTX
RVSLP1  PLA
        STA   LABL1,Y
        INY
        CPY   LASTX
ENDLP1  BNE   RVSLP1
        LDA   LASTCH
        RTS
```

Cet exercice présente plusieurs points intéressants : l'utilisation d'instructions JSR et CALL, par exemple. Le sous-programme RVSWRD est analogue dans sa structure à celui de l'exercice 1, mais il n'inverse que les caractères d'un mot, pas toute la chaîne. Dans les deux versions 6502 et Z80, le registre d'index (X et IX respectivement) est utilisé pour passer l'adresse de départ du mot au sous-programme, et l'accumulateur sert à repasser au programme principal la valeur du caractère qui a terminé le mot (soit un espace, soit le caractère terminateur de chaîne). C'est une technique très courante en langage d'assemblage, mais il faut l'utiliser avec précaution — surtout si on a l'habitude d'entrer sur la pile tous les registres CPU au début de chaque sous-programme (comme montré précédemment).

Une autre caractéristique importante est l'utilisation du registre Y dans la version 6502, d'abord pour garder l'adresse du mot pendant que X est utilisé comme index sur la boucle d'empilement, puis comme index sur la boucle de « désempilement » tandis que X garde l'adresse de fin du mot. « Adresse » est utilisé ici sans précision, car X et Y sont des registres à un octet, de sorte qu'aucun des deux ne peut comporter une adresse complète. A la place, ils contiennent dans ce cas un décalé de l'adresse LABL1. Par contraste, les registres d'index IX et IY de la version Z80 ne sont pas du tout employés; à leur place, on utilise les paires de registres HL et DE. Comme les registres X et Y de

```
                Z80
        ORG   $C000
LAST1   EQU   $0D
SPACE   EQU   $20
LABL1   DB    'CECI EST UN MESSAGE'
TERMN8  DB    LAST1
;
BEGIN   LD    DE,LABL1-1
LOOP0   CALL  RVSWRD
        CP    LAST1
ENDLP0  JR    NZ,LOOP0
        RET
;
;***SP   INV.  D'UN MOT ***
LASTCH  DB    $00
RVSWRD  PUSH  DE
        POP   HL
        INC   HL
RVSLP0  INC   DE
        LD    A,(DE)
        PUSH  AF
        CP    SPACE
        JR    Z,CLRSTK
        CP    LAST1
ENDRV0  JR    NZ,RVSLP0
CLRSTK  POP   AF
        LD    (LASTCH),A
;
RVSLP1  POP   AF
        LD    (HL),A
        INC   HL
        LD    A,L
        CP    E
        JR    NZ,RVSLP1
        LD    A,H
        CP    D
ENDRV1  JR    NZ,RVSLP1
        LD    A,(LASTCH)
        RET
```

6502, ceux-ci contiennent les adresses de début et fin de mot, mais au lieu d'être des index sur une adresse de base, ils sont utilisés comme adresses indirectes (l'instruction LD A,(DE) signifie « charger l'accumulateur à partir de l'octet dont l'adresse est en DE »). Toutes les paires de registres Z80 peuvent être utilisées de cette façon. Le jeu d'instructions est limité par le manque d'instruction pour comparer deux octets. Ainsi, pour comparer DE et HL, il faut d'abord comparer E à L, puis D à H. De même, dans la version 6502, X et Y sont comparés indirectement en utilisant un emplacement de mémoire, puisqu'il n'y a pas d'instruction pour comparer X à Y.

Le basic français

Vous vous êtes peut-être souvent demandé pourquoi les programmes basic utilisaient toujours des expressions en anglais. Si vous voulez du basic en français, en voici.

Il est désormais possible en France de programmer son micro-ordinateur dans sa langue maternelle. Venu du Canada, le BASIC français fonctionne sur Apple II+, Apple IIe et compatibles. Ce langage suit exactement les mêmes règles et la même syntaxe que le BASIC original, et l'on s'en sert pour programmer de la même façon. Il suffit de substituer à chaque terme du BASIC anglais son équivalent en BASIC français. Celui-ci permet également de traduire les instructions d'un programme initialement écrit en BASIC anglais.

Ces facilités permettent aux débutants, et à tous ceux pour lesquels la langue anglaise constitue un obstacle psychologique, de programmer plus commodément et plus efficacement.

Ainsi, on peut retrouver en français les quatre structures de base de toute programmation : la séquence, le choix, la répétition, la routine.

La séquence s'exprime de la même manière, puisqu'elle consiste en une numérotation des lignes.

Le choix s'exprime par le mot « SI » suivi d'une condition, par exemple :

```
20 SI A < 5 ET B >8 ALORS 40
```

La répétition ou la boucle se retrouve sous la forme :

```
RÉPÈTE X=1 JUSQUE 10
ENCORE X
```

La routine, c'est-à-dire une série d'instructions accomplissant une tâche ou une série de tâches particulières, à laquelle on peut faire appel plusieurs fois, prend deux formes en français :
— celle des instructions « VAVIENS » et « REVIENS », permettant d'écrire des sous-programmes qui seront exécutés chaque fois que nécessaire;
— celle qui permet, par l'instruction « DEF FN » de calculer une valeur au moyen d'une formule complexe, telle que :

```
10 DEF FN FONCT(BIDON)=BIDON*BIDON+2*BIDON+1
```

Voici un exemple de petit programme écrit en BASIC français :

```
10 DIM CUBE(10)
20 RÉPÈTE X = 1 JUSQUE 10
30 CUBE(X) = X * X * X
40 ENCORE X
50 DEMANDE "NOMBRE ? ";Y
60 SI Y < 1 OU Y >10 VATEN 90
70 ÉCRIS "LE CUBE DE ";Y;" EST ";CUBE(Y)
80 VATEN 50
90 FIN
```

Souhaitons que cette sympathique initiative permette à tous les enfants et à tous ceux qui ne connaissent pas l'anglais d'approfondir la logique, seul élément indispensable pour programmer, et ensuite de perfectionner son talent en cette matière nouvelle du XXIe siècle.

Liste des principales instructions et fonctions du basic français :

ARR : arrêt du programme.
ATT : attend, dans une pause conditionnelle.
CAPTE : permet de saisir un seul caractère au clavier sans l'afficher à l'écran.
CH : charge en mémoire un programme.
CLIN : provoque le clignotement des caractères affichés.
COUL : permet de choisir la couleur désirée en mode graphique basse résolution.
DEF : permet de définir ses propres fonctions.
DEMANDE : permet d'entrer des données à partir du clavier.
DIM : permet de définir un ou plusieurs tableaux.
DRT$() : permet d'extraire d'une chaîne de caractères les *n* caractères les plus à droite.
DESS : permet d'afficher à l'écran un petit pavé aux coordonnées voulues.
ÉCRIS : permet d'afficher des nombres ou des chaînes de caractères.
EFF : permet d'effacer l'écran.
ENCORE : cette instruction marque la fin d'une boucle : RÉPÈTE...JUSQUE...ENCORE.
ENR : enregistre un programme qui se trouve en mémoire.
ENT() : fonction qui calcule l'entier inférieur le plus près du nombre donné.
ET : opérateur logique.
FAIS : provoque l'exécution d'un programme.
GCH$() : permet d'extraire d'une chaîne de caractères les *n* caractères les plus à gauche.
HTAB : cette instruction place le curseur dans la énième colonne de l'écran.
INFO : permet de créer une liste de données qui pourront par la suite être lues par l'instruction "VOIS".
JUSQUE : précède la seconde limite d'une boucle "RÉPÈTE...JUSQUE".
LISTE : fait défiler à l'écran le texte d'un programme emmagasiné en mémoire.
LONG( ) : fonction qui calcule le nombre de caractères dans une chaîne-texte.
MACH : appelle l'exécution d'un sous-programme en langage machine situé à l'adresse indiquée.
OTE : permet d'effacer d'un programme un bloc de lignes d'instructions consécutives.
OU : opérateur logique.
PAR : fixe le pas d'incrément dans une boucle.
PIGE : donne comme résultat le contenu de l'octet de mémoire à l'adresse donnée en paramètre.
PISTE : permet d'afficher, pas à pas, les numéros de lignes d'instructions d'un programme en cours d'exécution.
PLONGE : cette instruction permet de placer une valeur directement en mémoire.
RAZ : instruction qui remet tout à zéro, vide toutes les variables-texte et initialise tous les pointeurs.
RÉPÈTE : définit une boucle avec variable-compteur.
REVIENS : cette instruction met fin à un sous-programme.
SELON : instruction de branchement multiple.
SI : instruction conditionnelle devant être suivie d'une expression numérique.
TRACE : dessine une forme prédéfinie aux coordonnées indiquées en paramètre.
VAL() : fonction qui transforme en nombre une chaîne de caractères.
VATEN : instruction qui provoque un branchement dans l'exécution d'un programme.
VAVIENS : amène l'exécution d'un sous-programme au numéro de ligne donné en paramètre.
VOIS : instruction qui permet à l'ordinateur de lire une ou plusieurs données dans une liste INFO du programme.
VTAB : déplace le curseur à l'écran à la position verticale indiquée en paramètre.

PROGRAMME N° 22

PROGRAMME DE TRI

Rappelez-vous notre programme n° 19. Nous avions introduit la notion de « table », et présenté l'édition et le traitement d'une table de *n* éléments. Nous supposons maintenant que l'on veuille trier ces éléments.
Plusieurs méthodes sont utilisables. Nous en présentons une qui n'est probablement pas la meilleure sur le plan de la rapidité, mais qui a l'avantage d'être très simple.
Commençons d'abord par rechercher la meilleure note. Nous supposons d'abord que la première note est la meilleure, c'est-à-dire la plus élevée. Ensuite, nous comparons cette note avec la suivante. Si celle-ci est supérieure, elle devient alors la meilleure note, etc.

```
        VA ( )
VA (1)  11
VA (2)  17
VA (3)  12
        14
         3
```

ME (meilleure note) vaut 11, au début, puis après la comparaison avec VA (2), ME sera égal à 17, etc.

Voici le programme correspondant :

```
5   REM ON DIMENSIONNE LA TABLE
10  DIM VA(5)
20  REM SAISIE DES NOTES
30  FOR I = 1 TO 5
40  INPUT "NOTE ?:";VA(I)
50  NEXT I
55  REM ON INITIALISE ME
60  ME = VA(1)
65  REM BOUCLE DE COMPARAISON
70  FOR I = 2 TO 5
80  IF VA(I) > ME THEN ME = VA(I)
90  NEXT I
95  REM  ON ECRIT LA PLUS GRAND VALEUR TROUVEE
100 PRINT : PRINT : PRINT "LA PLUS GRANDE VALEUR EST :";ME
```

```
NOTE ?:12
NOTE ?:15
NOTE ?:9
NOTE ?:5
NOTE ?:16


LA PLUS GRANDE VALEUR EST :16
```

```
NOTE ?:10
NOTE ?:6
NOTE ?:18
NOTE ?:11
NOTE ?:8


LA PLUS GRANDE VALEUR EST :18
```

Remarque. Ce petit programme ne nous permet pas de trier la table VA ( ), mais simplement de rechercher la note la plus élevée de VA ( ). Essayons maintenant de stocker non seulement la meilleure note, mais aussi la position de cette meilleure note dans les tables, et également de retrouver le nom de l'élève correspondant à cette note.

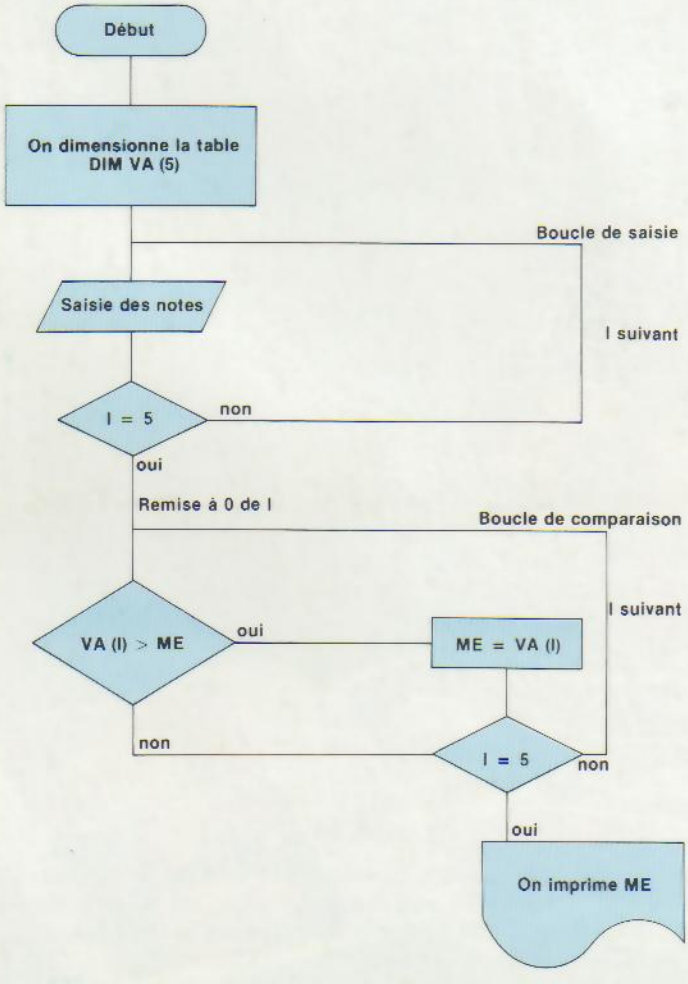
```
        VA ( )  EL$ ( )

VA (1)  11      DUPONT
VA (2)  17      DE ZAN
VA (3)  12      PALIX
        14      BRULE
         3      EMELINA
```

ME prendra la valeur 17; le rang de ME dans la table VA sera donc 2. Ainsi, on pourra lire le nom de l'élève concerné dans la table EL$, ici DE ZAN.

```
5     REM  ON DIMENSIONNE VA
10    DIM VA(5)
15    REM  ON DIMENSIONNE EL$
20    DIM EL$(5)
25    REM  BOUCLE DE SAISIE DES NOTES ET NOMS DES ELEVES
30    FOR I = 1 TO 5
39    PRINT
40    INPUT "VALEUR ?:";VA(I)
50    PRINT
60    INPUT "NOM DE L'ELEVE ?:";EL$(I)
70    NEXT I
75    REM  INITIALISATION DU TEMOIN DE POSITION
80    PM = 1
85    REM  BOUCLE DE COMPARAISON
90    FOR I = 2 TO 5
100   IF VA(I) > VA(PM) THEN PM = I
110   NEXT I
115   REM  IMPRESSION DE LA PLUS GRANDE VALEUR ET DU NOM
      DE L'ELEVE CONSIDERE
120   PRINT : PRINT : PRINT "LA MEILLEURE NOTE EST :";VA(PM)
130   PRINT : PRINT "OBTENUE PAR L'ELEVE :";EL$(PM)
```

```
VALEUR ?:11

NOM DE L'ELEVE ?:DUPONT

VALEUR ?:17

NOM DE L'ELEVE ?:DE ZAN

VALEUR ?:12

NOM DE L'ELEVE ?:PALIX

VALEUR ?:14

NOM DE L'ELEVE ?:BRULE

VALEUR ?:3

NOM DE L'ELEVE ?:EMELINA


LA MEILLEURE NOTE EST :17

OBTENUE PAR L'ELEVE :DE ZAN
```

ORGANIGRAMME



ORGANIGRAMME

VA représente valeurs, et EL$ la table élèves.